CPSC 320 2024W1: NP-Completeness Tutorial Problems

1 Scheduling classes

A college wishes to offer evening courses and has n possible courses that can be offered. There are k students interested in taking a course and each student is interested in some subset of the n possible courses. To manage costs, the college has a bound b on the number of courses it is willing to actually offer. Is there a way to choose b courses out of the n possibilities so that every student can take a course that interests them? Formally, an instance of the CS problem consists of

- a set C of n possible courses, numbered 1 through n,
- subsets $S_1, S_2, \ldots, S_k$ of $\{1, 2, \ldots, n\}$ (the courses of interest to each of the students), and
- a bound b (the maximum number of classes actually offered).

The problem is to determine whether there is a subset B of $\{1, 2, \ldots, n\}$, where the size of B is $\leq b$, such that $B \cap S_i$ is not empty for all i, $1 \leq i \leq k$.

1. Show that the CS problem is in NP.

   A certification algorithm takes as input an instance $I = (n, C, S_1, \ldots, S_k, b)$ as well as a certificate (or potential solution), which is a subset B of $\{1, 2, \ldots, n\}$. The certification algorithm checks that (i) the size of B is at most b, and (ii) that $B \cap S_i$ is not empty for all $i, 1 \leq i \leq k$. If either check fails, the algorithm rejects, otherwise the certification algorithm accepts.

   The algorithm time to check the size of B is $O(n)$, and the time to check that $B \cap S_i$ is not empty should also be $O(n)$ if we assume that each S_i and also B is sorted. k such checks are needed, so the total time is $O(kn)$, plus possibly the time to sort each of the sets, which in total could be $O(kn \log n)$. This is polynomial in n and k.

   If I is a yes-instance of CS, then there is a certificate that causes the certification algorithm to output yes, and if I is a no-instance then on every certificate, one of the checks will fail and the algorithm rejects. Therefore the certification algorithm is correct.

   In summary, we have a correct certification algorithm for CS that runs in polynomial time, so CS is in NP.

2. Describe an efficient reduction from the Vertex Cover problem to the Class Scheduling (CS) problem. The Vertex Cover problem is: Given an undirected graph $G = (V, E)$ and an integer K, does G have a vertex cover with size at most K? A vertex cover is a subset W of V such that every edge in E has at least one endpoint in W.

   Let (G, K) be an instance of the Vertex Cover problem. We map $(G = (V, E), K)$ to an instance of CS as follows. Assume that the set $V = \{1, 2, \ldots n\}$ and let the edges of E be $e_1, e_2, \ldots, e_m$. The set C of courses is the set V of vertices. There is one student per edge in E. The set S_i of courses that interest student e_i in are the two endpoints of e_i in the graph G. Finally, the bound b is set to equal K. The mapped instance is $(V, S_1, S_2, \ldots, S_m, b)$.

3. Your reduction maps instances (G, K) of Vertex Cover to instances $(C, S_1, \ldots, S_k, b)$ of CS. Show that G has a vertex cover of size at most K if and only if in the mapped instance $(C, S_1, \ldots, S_k, b)$, there is a set of courses of size at most b such that every student can take a course that interests them. (Remember that you need to two parts here, one for "if" and one for "only if".)

   If direction: Suppose that $(V, S_1, S_2, \ldots, S_m, b)$ is a "yes" instance of CS. Then there is a set B of courses that satisfies all students, i.e., such that one course is in each set S_i, and the size of B is at most b. Since $B \subseteq V$, B corresponds to a set of nodes of G, and since there is one student per edge of G and all students are statisfied, B must cover all edges in G. Therefore G has a vertex cover of size $b = K$ and so (G, K) is a "yes" instance of vertex cover.

   Only if direction: Suppose that (G, K) is a "yes"-instance of VC. In this case, there is a set of nodes, call it B, which is of size at most K that covers all edges of the graph. That is, every edge of the graph has an endpoint in B. Therefore, in the mapped instance, B is a set of courses that satisfies all students, since students correspond to edges of G. Also, B has size at most b, since $b = K$.

4. Explain why your reduction runs in polynomial time.

   This reduction can be computed in time $O(n+m)$, simply by scanning the instance (G, K) to produce the corresponding components of the instance $(V, S_1, S_2, \ldots, S_m, b = K)$.

5. Put the previous parts together to conclude that CS is NP-complete.

   We have shown that CS is in NP, and have also shown that Vertex Cover $\leq_p$ CS. Therefore CS is NP-complete.

2 NP True or False

Let X and X' be decision problems, where both problems have Yes instances and No instances. State whether you think each of the following statements must be true, must be false, or is an open question. Justify your answer.

1. **Statement:** If $X \leq_p X'$ and X is *not* in NP, then X' is not in NP.

   True. We show that if X' were in NP, then X must also be in NP, getting a contradiction. Let f be a polynomial-time reduction from instances of X to instances of X'. Since we assume that X' is in NP, there must be an efficient certifier for X', let's call this algorithm Certifier-X'.

   To show that X is in NP, we construct an efficient certifier for X. This certifier takes an instance I and a certificate, say C', computes $f(I)$, and then runs algorithm Certifier-X' on $(f(I), C')$. If I is a Yes-instance of X, then I' is a Yes-instance of X', and so there is some polynomial-size certificate that causes Certifier-X' to output Yes. Also, if I is a No-Instance of X, then $f(I)$ is a No-instance of X', and Certifier-X' outputs No, no matter what the certificate is.

2. **Statement:** If $X \leq_p X'$, X is in P and X' is in NP, then X' must be in P.

   This is an open question. It is relatively easy to reduce an instance of an easy problem X (say, interval scheduling, which can be interpreted as finding a maximum independent set in a graph that is an instance of a special type of graphs called *interval graphs*) to an instance of an NP-complete problem X' (say, finding a maximum independent set in a graph). In fact, the reduction can first efficiently solve the instance of X, and then, depending on whether the instance is a Yes-instance or No-instance, map it to either a trivial "Yes" instance or a trivial "No" instance of the hard problem. This is possible since X' contains both types of instances.

   So, even if X is in P, it might be that X' is NP-complete. If we knew that P $\neq$ NP, we could conclude that statement 2 is false. However, it is an open question whether P $=$ NP: even if X' is NP-complete, it's conceivable that X' is in P. (Many people think that it is unlikely that P $=$ NP but we don't know for sure.) So the statement above is an open question.

   Another point to take away from this is that if you reduce a problem X (which is in NP) to an NP-complete problem X', this does NOT tell you that X is NP-complete. You have to reduce X' to X to show that X is NP-complete. This is counterintuitive at first and even trips up people with experience. Make sure you understand which direction the reduction should go, and why, and pause to think before starting on the details of a reduction.