CSCI3390-Second Test with Solutions

April 26, 2016

Each of the 15 parts of the problems below is worth 10 points, except for the more involved 4(d), which is worth 20. A perfect score is 100: if your score is in excess of 100, I will treat the additional points as extra credit to be added to either homework or test scores. These are independent of one another, so you can work them in any order, but make sure you label clearly which question you are answering!

1. Let Σ be a finite alphabet. Let $L \subseteq \Sigma^*$, where $L \in \mathbf{NP}$. Answer the following questions 'True', 'False', 'Believed True', or 'Believed False'. The last two categories are to be used if the answer depends on the truth of a widely-believed, but not-yet-proved, conjecture.

   (a) Every such L is Turing-recognizable.

   Solution. True, because every $L \in \mathbf{NP}$ is decidable (see (d)).

   (b) Every such L is decided by a deterministic TM that takes $O(n^k)$ steps on inputs of length n. (Here and in the subsequent parts $k > 0$ is a constant that depends on the TM but not on n.)

   Solution. Believed False—this is the definition of **P**, so the statement says $\mathbf{NP} \subseteq \mathbf{P}$.

   (c) **Some** such L is decided by a deterministic TM that takes $O(n^k)$ steps on all inputs of length n.

   Solution. True, this says **NP** contains some problems in **P** (in fact it contains every problem in **P**).

   (d) Every such L is decided by a deterministic TM that takes $O(2^{n^k})$ steps on all inputs of length n.

Solution. True. This is a precise statement of what we mean when we say that every problem in **NP** has a brute-force solution that consists of trying out every possibility. The TM systematically enumerates all possible 2^{n^r} guesses of n^r bits, and on each guess runs for n^ℓ steps. The total run-time is no more than

$$n^\ell \cdot 2^{n^r} < 2^n \cdot 2^{n^r} = 2^{n+n^r} < 2^{n^{r+1}}.$$

Of all the parts of this problem, this is the one that was most frequently answered incorrectly.

2. Let $COMPOSITE \subseteq \{0, 1, \ldots, 8, 9\}^*$ be the set of decimal representations of composite positive integers. That is, L is the set of strings

$$\{4, 6, 8, 9, 10, 12, 14, 15, \ldots\}.$$

(a) Give a simple proof, involving no special facts from number theory, that $COMPOSITE \in \mathbf{NP}$.

Solution. The corresponding verifier problem is: Given integers N and k, do we have $k|N$ with $k \neq 1$ and $k \neq N$? We can do the division in time quadratic in the number of digits of N and linear in the number of digits of N, and the additional information (k) has no more digits than N, so this is a polynomial-time verifier.

(b) The following is a 'proof' that $COMPOSITE \in \mathbf{P}$: Given input N, divide N in turn by $2, 3, \ldots, N-1$. Answer 'Yes' and halt if the remainder is zero, and answer 'No' otherwise. Since division of two integers takes quadratic time, and we perform at most $N - 2$ divisions, this algorithm takes time polynomial in the size of the input. This argument is incorrect—what is the error?

Solution. The size of the input is the number of digits of N, not N itself, so this algorithm takes exponential time in the size of the input.

(c) So, is $COMPOSITE$ in **P**? You don't have to prove your answer, you can simply cite theorems that were discussed in class.

Solution. As mentioned in class $PRIMES \in \mathbf{P}$, and $\mathbf{P}$ is closed under complement, so we now know that $COMPOSITE \in \mathbf{P}$ as well. It was common to answer this incorrectly by citing an algorithm (like using Fermat's Theorem, or Miller-Rabin) that does not really show $PRIMES \in \mathbf{P}$. The correct answer should cite the AKS algorithm.

3. These problems concern boolean satisfiability.

   (a) Consider the following propositional formula:

   $$(p \vee q \vee r) \wedge (\neg p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge (\neg q \vee \neg r) \wedge (\neg q).$$

   Is this formula satisfiable? If so, give *all* satisfying assignments. (It may helpful to note that the first four clauses exhibit a particular pattern that we've seen before.)

   Solution. The first four clauses say that exactly one of p, q, r is true, and the last that q is false, so there are exactly two satisfying assignments: Setting p true and q, r false; and setting p, q false and r true.

   (b) In class we showed that 2-SAT is in **P**. The formula above is not a legal input to 2-SAT, since the first clause contains more than two literals. Show nonetheless, satisfiability for propositional formulas that contain *at most one* clause with more than 2 literals is in **P**. (HINT: You don't have to tell me the algorithm for 2-SAT, but you need to call this algorithm.)

   Solution. Our problem has the form $\theta \wedge \phi$, where ϕ is a 2-CNF formula, and θ is a disjunction of literals. We can test each formula $\ell \wedge \phi$, where ℓ is a literal in θ separately for satisfiability, and answer yes if any of these is satisfiable. Each such formula is a 2-CNF, so each test takes polynomial time, and the number of tests is equal to the size of θ, thus smaller than the size of the original formula, so this takes polynomial time.

   Some students answered this by saying, in effect, try out all satisfying assignments for ϕ and test if any of them is satisfied by θ, but there may be too many satisfying assignments for θ to do this in polynomial time.

   (c) A boolean formula is a *tautology* if **every** assignment of truth values to variables is a satisfying assignment. Is the problem of determining whether a boolean formula in CNF is a tautology in **P**? Does the answer depend on the status of the $\mathbf{P} = \mathbf{NP}$ conjecture?

   Solution. This is in **P** unconditionally. We require that each clause separately be a tautology, which happens if and only if the clause contain both p and $\neg p$ among its literals. We can check for this with a single scan of the formula. (This is very similar to a homework problem about satisfiability for DNF formulas.)

(d) A boolean formula is a *contradiction* if **no** assignment of truth values to variables is a satisfying assignment. Is the problem of determining whether a boolean formula is a contradiction in **P**? Does the answer depend on the status of the $\mathbf{P} = \mathbf{NP}$ conjecture?

Solution. This problem is the complement of SAT. If $\mathbf{P} = \mathbf{NP}$, then $\mathbf{NP}$ is closed under complement, and thus this problem is in **P**. Conversely, if this problem is in **P**, then since **P** is closed under complement, we would bet $\mathbf{SAT} \in \mathbf{P}$ and thus $\mathbf{P} = \mathbf{NP}$. Some students got this far (and got full credit) but went on to say that $CONTRADICTION$ is **NP**-compete. This is (probably) not true, since $CONTRADICTION$ is **NP** is not believed to be closed under complement. However, it is **NP**-hard.

4. These problems concern the Hamiltonian Path problems for both directed and undirected graphs, and require you to correctly identify polynomial-time reductions. Here, to remind you, are the problems:

   DIRECTED HAMILTONIAN PATH

   Input: A directed graph G and two vertices s and t.

   Output: Yes if and only if there is a directed path from s to t that visits each vertex exactly once.

   UNDIRECTED HAMILTONIAN PATH

   The description is the same, except both the graph G and the required path are undirected.

   The first two parts of the problem involve the following two constructions on graphs:

   Construction 1:

   Starting from an undirected graph G, produce a directed graph G' with the same set of vertices by replacing each undirected edge $\{i, j\}$ by a pair of directed edges (i, j) and (j, i).

   Construction 2:

   Starting from a directed graph H produce an undirected graph H' with the same set of vertices by replacing each directed edge (i, j) where $i \neq j$ by the undirected edge $\{i, j\}$.

   The constructions are illustrated in the accompanying figure. Note that an undirected graph cannot have multiple edges between the same two vertices, nor loops at a single vertex, so if H contains a pair of edges (i, j) and (j, i), H' will only have $\{i, j\}$, and if H contains loops (i, i), these will not appear in H' at all.

   (a) Choose which one of the following answers is appropriate, and fill in the blanks:

   > Construction 1 is a polynomial-time reduction from [PROBLEM] to [PROBLEM]. If we already know [PROBLEM] is **NP**-hard, this reduction proves [PROBLEM] is $\mathbf{NP}$-hard.

   OR

Construction 1 is not a polynomial-time reduction from either of these problems to the other because.....

Solution. Construction 1 is a polynomial-time reduction from **UNDIRECTED HAMILTONIAN PATH** to **DIRECTED HAMILTONIAN PATH**. It shows that if **UNDIRECTED HAMILTONIAN PATH** is **NP**-hard, then **DIRECTED HAMILTONIAN PATH** is **NP**-hard. Observe that in class we proved that these problems were **NP**-hard in the opposite order, first showing that the directed problem was **NP**-hard.

(b) Answer the above question for Construction 2.

Solution. This is not a reduction in either direction. Since it converts a directed graph into an undirected graph, if it were a reduction at all, it would have to be from **DIRECTED HAMILTONIAN PATH** to **UNDIRECTED HAMILTONIAN PATH**. But it does not preserve the Hamiltonian Path property: Observe that in the diagram, the undirected graph at bottom has a Hamiltonian path from 1 to 3, but there is no such Hamiltonian path in the directed graph.

(c) The *degree* of a vertex in an undirected graph is the number of neighbors it has. (See the caption of the accompanying figure.) Prove that if we restrict to graphs in which every vertex has degree at most 2, then **UNDIRECTED HAMILTONIAN PATH** is in **P**.

Solution. In a graph in which every vertex has degree no more than 2, every connected component is either a loop or a straight line, and thus if there is a Hamiltonian path from s to t, G must be connected, and either s is on one end of a line and t is on the other, or s and t are adjacent vertices in a loop. We thus get the following algorithm: Start at s. If s has two neighbors, and one of the neighbors is not t, then reject. Otherwise, visit the other neighbor, and continue visiting unvisited adjacent neighbors until you can go no further. If the last vertex is t and all vertices of the graph have been visited, accept, otherwise reject. If s has only one neighbor, visit it, and continue visiting unvisited adjacent neighbors until you can go no further. If the last vertex is t and all vertices of the graph have been visited, accept, otherwise reject. This is just breadth-first search applied in the very special case of graphs with maximum degree 2, and is thus a polynomial-time algorithm (in fact linear in the number of vertices).

(d) (Harder.) What if we restrict to graphs in which every vertex has degree

at most 3? Show that this problem is **NP**-complete. (You may have to remember some proofs we did in class.)

Solution. Not just harder, but harder than I thought. If you go through the proof of the reduction of 3-SAT to **DIRECTED HAMILTONIAN PATH** you find that each vertex of the resulting directed graph has in-degree at most 3 and out-degree at most 3, and that the subsequent reduction to **UNDIRECTED HAMILTONIAN PATH** has degree at most 4 at every vertex. So this shows that **UNDIRECTED HAMILTONIAN PATH** for graphs with maximal degree 4 (not 3, as I first thought) is **NP**-hard. The claim is still true, and requires a subsequent conversion to graphs with maximal degree 3 that preserves the Hamiltonian path property, which I'm still thinking about!.

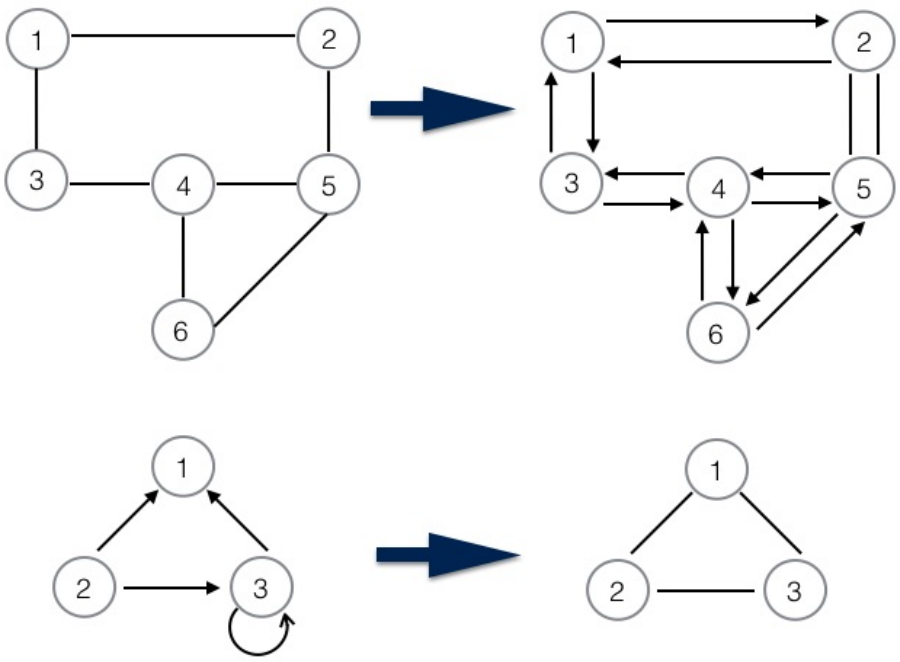

Figure 1: Construction 1 (top) and construction 2(bottom). In the top left diagram, vertices 1,2,3 and 6 all have degree 2, vertices 4 and 5 have degree 3. All vertices in the lower right have degree 2.