15-251: Great Theoretical Ideas In Computer Science

Recitation 7 Solutions

Proving that languages are in NP is not hard...

(a) A composite number is a number that is not a prime number and is not 1. Let COMPOSITE be the following decision problem: Given as input a positive integer N written in binary, is N composite? Prove that COMPOSITE $\in$ NP.

Let our certificate be a pair of positive natural numbers.
Our verifier first checks that neither number is equal to 1.
It then multiplies the numbers together, and checks equality to the original input.
This verifier is in polynomial time, as multiplication and equality checking are polynomial time.
Any composite number n has two such numbers that multiply together for n (factors).
Further, these two numbers are polylength, as they must each be less than or equal to n.
Conversely, having two such numbers is the definition of being composite.
It follows that $COMPOSITE \in NP$.

(b) Show that any language in NP can be decided in exponential time.

Let our NP language be L, and let V be the verifier for L. Let the running time of V be $O(n^k)$ for some fixed k (where n is the length of the input to V).
Suppose that for any $x \in L$, the 'proof' of the fact that $x \in L$ has length at most $O(n^p)$ for some fixed p (we know such a p exists because $L \in$ NP).
The goal of this problem is to show that every language in NP can be decided by brute-force search. The following strategy works : for any input y, enumerate all possible proofs of length smaller than $O(|y|^p)$, and run V on each proof (along with y). If V ever accepts, $y \in L$, otherwise $y \notin L$.
Since V is a polynomial time Turing-machine, this takes exponential time overall (because the number of proofs that we need to check is exponential in y).

Properties of bipartite graphs

Show that G is bipartite if and only if it contains no cycles of odd length.

Claim 1. Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. Then G has no odd cycles.

Proof. Suppose $v_1, v_2, ..., v_n, v_{n+1} = v_1$ is a cycle in G. Without loss of generality, assume $v_1 \in V_1$. Since $\{v_1, v_2\} \in E$ and G is bipartite, v_2 must be in V_2. A simple induction argument extends this to all v_i: if i is odd, then $v_i \in V_1$, and if i is even, then $v_i \in V_2$. Since $v_{n+1} = v_1$ it is in V_1, so $n+1$ must be odd, and hence n (the length of the cycle) is even.

Claim 2. Let $G = (V, E)$ be a graph with no odd cycles. Then G is bipartite.

Proof. First, observe that if G is disconnected and G_1, G_2, ..., G_k are the connected components of G, then G is bipartite if and only if each G_i is bipartite. So we may assume that G is connected.

Now, choose an arbitrary vertex $r \in V$. Partition the vertices of V into sets $X_0, X_1, X_2, ..., X_k$,

where for $1 \leq i \leq k$, the vertices v in X_i are those whose distance from r, $d_r(v)$, is i. For example, X_0 is $\{r\}$. Now we show that if $\{x, y\}$ is an edge, then $|d_r(x) - d_r(y)| = 1$. First, notice that $|d_r(x) - d_r(y)| \leq 1$: if $d_r(x) > d_r(y)$, then there is a path from r to x of length $d_r(y) + 1$ formed by taking a path from r to y of length $d_r(y)$ and then following the edge $\{x, y\}$.

So assume for contradiction that $d_r(x) = d_r(y)$. Call this k. Let $r = a_0, a_1, ..., a_k = x$ be a shortest $r - x$ path, and let $r = b_0, b_1, ..., b_k$ be a shortest $r - y$ path. Let $0 \leq i < k$ be the largest index such that $a_i = b_j$, for some $0 \leq j < k$. Since $a_i = b_j$, they must be at the same depth, so in fact i is equal to j. Now we have a cycle $a_i, a_{i+1}, ..., a_k = x, y = b_k, b_{k-1}, ..., b_i$. This has $2(k - i) + 1$ vertices, which contradicts the assumption that G has no odd cycles. Therefore, $|d_r(x) - d_r(y)| = 1$.

Finally, we can construct a bipartition of G. Let

$$V_1 = \{v \in X_i \mid i \text{ is even and } 0 \leq i \leq k\}$$
$$V_2 = \{v \in X_i \mid i \text{ is odd and } 0 \leq i \leq k\}$$

From the fact that $|d_r(x) - d_r(y)| = 1$ for all edges $\{x, y\}$, this is a bipartition.

Assorted reductions between graph problems

HAMILTONIAN-CYCLE is the following problem: Given an undirected graph, is there a cycle that visits every vertex exactly once? HAMILTONIAN-CYCLE is NP-complete. Show that the following problems are NP-complete:

(a) Given a graph G, is there a path that visits every vertex of the graph exactly once?

(b) Given a graph G and an integer k, is there a path of length k?

(c) Given a graph G and an integer k, is there a spanning tree in G that contains at most k leaves?

(a) This is the HAMILTONIAN-PATH problem. To show it is NP-Complete we show it is both in NP and NP-Hard.

Our verifier takes in a graph (V', E') and checks that it is indeed a path that visits all vertices.

We then show it is in NP-Hard by reducing it from the problem of HAMILTONIAN-CYCLE. First consider the following method to check if a graph has a Hamiltonian cycle containing a given edge $\{u, v\}$: construct a modified graph G' by deleting $\{u, v\}$, creating a new vertex u' and connecting it only to u, and creating a new vertex v' and connecting it only to v. Now if G' has a Hamiltonian path (which we can check with one call to the HAMILTONIAN-PATH oracle), note that the path must start with u', u and end with v, v' (or vice versa). If we cut off the endpoints u', v' and re-add the deleted edge $\{u, v\}$, then we have a Hamiltonian cycle in the original graph, one of whose edges is $\{u, v\}$. Conversely, if there was a Hamiltonian cycle containing $\{u, v\}$ in the original graph, our oracle call will pick up on the Hamiltonian path in G' that is formed by connecting u', v' as endpoints to the Hamiltonian path from u to v in the original graph.

So, given an edge, we can know whether or not our graph contains a Hamiltonian cycle containing that edge. Now just pick some vertex t and use the method described above on all edges incident to t: if any of the checks succeed, then we are done because our graph does contain a Hamiltonian cycle. If none of them succeed, we know that our graph has no

Hamiltonian cycle, as a Hamiltonian cycle would have to visit t at some point and therefore use an edge incident to it.

Since it is both in NP and NP-hard, HAMILTONIAN-PATH is NP-Complete.

(b) Call the problem PATHK. To show it is NP-complete we show it is both in NP and NP-Hard.

Our verifier takes in a graph (V', E') as a proof. It checks that indeed forms a path of length k. The naive algorithm runs in polynomial time.

We then show it is NP hard by reducing it from the problem of HAMILTONIAN-PATH. We note that a Hamiltonian path is equivalent to a path that visits all vertices of a graph, thus it is a path of length $|V| - 1$. So given an instance of HAMILTONIAN PATH $G = (V, E)$, it is equivalent to the problem of PATHK$(G, |V| - 1)$.

Since it is both in NP and NP-Hard, it is NP-Complete.

(c) Call the problem SPANK. To show it is NP-Complete we show it is both in NP and NP-Hard.

Our verifier takes in a graph (V', E') as proof. Using any of our favourite graph search algorithms, say, BFS, check that it is indeed a spanning tree and has at most k leaves.

We then show it is NP hard by reducing it from the problem of HAMILTONIAN-PATH. We note that a Hamiltonian path is a spanning tree by definition with at most 2 leaves (accounting for the trivial edge cases of the empty graph and the graph on one node). Conversely, a spanning tree with at most 2 leaves is a also Hamiltonian path. So given an instance of HAMILTONIAN-PATH G, it is equivalent to the problem of SPANK $(G, 2)$.

...but proving that they're NP-complete is not easy.

The president of a large country can afford to build hospitals in up to k different towns. The goal is that everybody has a hospital in their town, or at least in a neighboring town. Show that determining if this is possible is NP-complete. More formally, let

$$\text{HOSP} = \{\langle G = (V, E), k\rangle : \exists H \subseteq V \text{ with } |H| \leq k \text{ such that } \forall v \in V, \text{ either } v \in H \text{ or } w \in H \text{ for some } \{v, w\} \in E\}.$$

Show that HOSP is NP-complete.

The NP-complete problem called VERTEX-COVER might be useful for this (we suggest that you reduce VERTEX-COVER to HOSP). VERTEX-COVER is defined as follows :

$$\text{VERTEX-COVER} = \{\langle G = (V, E), k\rangle : \exists C \subseteq V \text{ such that } |C| = k \text{ and } \forall u, v \in E,\ (u \in C) \vee (v \in C)\}$$

Informally, the problem can be described as follows : given a graph G and a natural number k, is it possible to color exactly k vertices red such that every edge has at least one vertex colored red?

To prove that HOSP is NP-complete, we must prove both that it is in NP and NP-hard. It is easy to see that it is in NP: if there is a solution, the a set of hospitals H is a polynomial-length certificate. To verify this certificate, just check for each vertex if either it is in H or one of its neighbors is.

To show that HOSP is in NP-hard, we reduce VERTEX-COVER to HOSP (in polynomial time). Let $(G = (V, E), k)$ be an instance of VERTEX-COVER. If $k > |V|$, we can output YES immediately. Otherwise, construct an input to HOSP as follows. First, remove all of the isolated vertices from V. For each edge $e = \{x, y\} \in E$, add a vertex v_e and edges $\{v_e, x\}$ and $\{v_e, y\}$. Call this graph G', and return HOSP(G', k). Since we added only $|E|$ vertices and $2|E|$ edges, this is a polynomial-time reduction. It remains to prove the following claim.

Definition. Let H be a subset of the vertices of some graph F. We say that H covers a vertex v if either v is in H or v is adjacent to a vertex in H. If every vertex in F is covered by H, then we say H is a hospital cover of F.

Claim. For any graph $G = (V, E)$, let the graph that we construct be $G' = (V', E')$, and let $k \leq |V|$. Then G' has a hospital cover of size at most k if and only if G has a vertex cover of size k.

Proof. Suppose G has a vertex cover $H \subseteq V$ of size k. Let V_0 be the set of isolated vertices in G. Let $H' = H \setminus V_0$: we will show that H' is a hospital cover of G'. Let $v \in V' \cap V$. By the construction, v is incident with some edge $\{u, v\} \in V$. Since H is a vertex cover, at least one of u and v is in H, so v is covered by H. Now, let $v \in V' \setminus V$ instead. By the construction, v is equal to $v_{\{x,y\}}$ for some $\{x, y\} \in E$. Since x and y are not isolated, at least one of x and y is in H', so v is covered by H. Therefore H' is a hospital cover of G' of size $|H'| \leq |H| = k$.

Next, suppose that G' has a hospital cover H' of size at most k. Define a set $H \subseteq V$ of size k as follows. Replace each vertex $v_{\{x,y\}} \in H' \setminus V$ with either x or y, arbitrarily. Then, add $k - |H'|$ more vertices of V to H (since $k \leq |V|$, there are enough vertices in V to do this). It remains to show that H is a vertex cover of G.

Let $e = \{x, y\} \in E$. If either x or y was in H', then it is in H, so e is covered by H. If neither was in H', then v_e had no neighbors in H', so v_e must be in H'. By our construction of H, either x or y must therefore be in H, so e is covered by H. Thus, H covers all of the edges of G, so G has a vertex cover of size k.

Thus, if HOSP has a polynomial time solution, then VERTEX-COVER does too. So, since VERTEX-COVER is in NP-hard, HOSP is too.