Sample Solution to Exam in Computational Complexity

2007/7/13

Oded Regev, Oded Schwartz, Amnon Ta-Shma

Disclaimer: Although we tried to be careful, these solutions might still contain a bug or two. If you find any, please let us know!

1. (a) First recall that NL = coNL. Therefore it is enough to show that D $\in$ coNL, or equivalently, that the problem of checking whether a given directed graph *contains* a directed cycle is in NL. To do so, we will now show a log-space verifier that reads its witness from a read-once tape (as we saw in class, NL can be defined using such verifiers).

   Our verifier expects to get as a witness a list of vertices that form a directed cycle in the graph. The verifier starts by saving the first vertex appearing in the witness into his work tape. Since one vertex takes only $\log n$ bits of space, he can do that. He then reads the rest of the witness, vertex by vertex, and each time verifies that the previously read vertex is connected to the currently read vertex by an edge. Notice that for this he only needs to remember the last vertex which takes $\log n$ bits. Finally, when he reaches the last vertex in the witness, he checks that this vertex is connected to the first vertex in the witness (which is stored in the work tape).

   It is easy to see that this verifier requires only $O(\log n)$ space, and that it uses only read-once access to the witness. Moreover, if the graph has a directed cycle then using the list of vertices on this cycle as a witness would make the verifier accept, and the size of this witness is polynomial; conversely, if the verifier accepts a witness, then the vertices on this witness must form a directed cycle in the graph (since each vertex is connected to the next one, and the last vertex is connected to the first one).

   (b) We will prove that D is NL-hard by showing a log-space reduction from the NL-complete language $\overline{\text{PATH}}$ to D. (Recall that $\overline{\text{PATH}}$ is the problem of deciding, given a directed graph G and two vertices s, t in it, whether there is *no* path from s to t in G. Since PATH is NL-complete, $\overline{\text{PATH}}$ is coNL-complete; but since NL = coNL, this means that $\overline{\text{PATH}}$ is NL-complete.)

   The reduction is as follows. Its input is a directed graph $G = (V, E)$ and two vertices $s, t \in V$. Let $n = |V|$ be the number of vertices in the graph. It outputs the graph G' constructed as follows. Its vertex set is $V \times \{1, \ldots, n\}$, of size n^2. For each $k = 1, \ldots, n-1$ and each edge $(i, j) \in E$ we add to G' an edge from (i, k) to $(j, k+1)$. In addition, for each $k = 1, \ldots, n$, we add to G' an edge from (t, k) to $(s, 1)$. Less formally, the graph G' is made of n layers where between any two consecutive layers we place a directed copy of the graph G with all edges going 'forward', and we in addition place 'backwards' edges from the node t in each layer to the node s in the first layer.

   It is easy to see that this reduction can be implemented in log-space. We now prove correctness. First, assume that there is a path in G from s to t, and assume its vertices are $(v_1 = s, v_2, \ldots, v_{k-1}, v_k = t)$. We can assume without loss of generality that the path is simple and so in particular $k \leq n$. We therefore see that $((v_1, 1), (v_2, 2), \ldots, (v_k, k), (v_1, 1))$ is a directed cycle in G', as required. Now assume that there is a directed cycle in G'. This cycle must clearly contain one of the 'backwards' edges (since without these edges

G' is acyclic). In particular, the cycle must contain the vertex $(s, 1)$. Following along the cycle from this vertex onwards, we encounter vertices $(s, 1), (v_2, 2), (v_3, 3), \ldots$ until at some point a backward edge must be used. Therefore this list of vertices must end with (t, k) for some k, and $(s, v_2, \ldots, v_{k-1}, t)$ forms a path from s to t in G.

Remark: Many students gave the following *incorrect* proof: Our goal is to show that $\bar{D}$ is NL-hard (which would imply that D is NL-hard since NL = coNL). To do this, we repeat the proof that PATH is NL-hard with a minor modification. Namely, let L be any language in NL and let M be a log-space NDTM for L. Our reduction from L to $\bar{D}$, given an input $x \in \{0,1\}^*$, outputs the graph obtained from the configuration graph $G_{M,x}$ by adding one extra edge from C_{accept} to C_{start}. The bug in this proof is that the configuration graph $G_{M,x}$ need not be acyclic (even though M always halts) since it might contain cycles that correspond to configurations that are unreachable from C_{start}.

2. (a) True. Assume that $L_1, L_2 \in$ NP. By definition, this implies that there are poly-time verifiers V_1, V_2 and polynomials p_1, p_2 satisfying that

$$\forall x \in \{0,1\}^*, \;\; x \in L_1 \iff \exists w \in \{0,1\}^{p_1(|x|)} \text{ s.t. } V_1(x, w) = 1,$$

$$\forall x \in \{0,1\}^*, \;\; x \in L_2 \iff \exists w \in \{0,1\}^{p_2(|x|)} \text{ s.t. } V_2(x, w) = 1.$$

In order to show that $L_1 \cap L_2 \in$ NP we now construct a poly-time verifier V for $L_1 \cap L_2$. The goal of V is to verify that its input x is in $L_1 \cap L_2$, i.e., both in L_1 and in L_2. It expects a witness of the form $\langle w_1, w_2 \rangle$. It first applies V_1 to the input x and the witness w_1, and then applies V_2 to the input x and the witness w_2. It accepts if and only if both calls accept. Clearly V runs in polynomial time, and it is a legal verifier since

$$\forall x \in \{0,1\}^*, \;\; x \in L_1 \cap L_2 \iff \exists w \in \{0,1\}^{p_1(|x|)+p_2(|x|)} \text{ s.t. } V(x, w) = 1.$$

(b) Equivalent to NP = coNP. To show this equivalence, we need to prove two things. First assume that NP = coNP. Then clearly if $L \in$ NP then L is also in coNP, which is (by definition) equivalent to saying that $\bar{L} \in$ NP.

Conversely, assume that for all $L \in$ NP it holds that $\bar{L} \in$ NP. Equivalently, this says that for all $L \in$ NP it holds that $L \in$ coNP. But this is exactly saying that NP $\subseteq$ coNP. Hence, to show that NP = coNP it is enough to show that also coNP $\subseteq$ NP. So let L be any language in coNP. By definition, its complement $\bar{L}$ is in NP, which, by our assumption, implies that $\bar{L} \in$ coNP and hence $L \in$ NP. So we see that coNP $\subseteq$ NP, and therefore NP = coNP as required.

(c) Also equivalent to NP = coNP. We again need to prove two things. First assume that NP = coNP. Then if $L_1, L_2 \in$ NP then since NP = coNP, we also have $\overline{L}_2 \in$ NP. Using (a), we have that $L_1 \setminus L_2 = L_1 \cap \overline{L}_2 \in$ NP.

Conversely, assume that for all $L_1, L_2 \in$ NP it holds that $L_1 \setminus L_2 \in$ NP. Take $L_1 = \{0,1\}^*$ to be the language of *all* strings. This language is clearly in NP (it is in fact regular). We obtain that for all $L \in$ NP, $\{0,1\}^* \setminus L = \bar{L}$ is also in NP, which, as we saw in (b), implies that NP = coNP.

3. (a) Our goal is to show a reduction that gets as input a graph $G = (V, E)$ and outputs a number m and sets $S_1, \ldots, S_n \subseteq \{1, \ldots, m\}$ such that: (a) if G has an independent set of size at least $\beta|V|$ then there are βn pairwise disjoint sets among $S_1, \ldots, S_n$ and (b) if the largest independent set in G is of size at most $\alpha|V|$, then there are at most αn pairwise disjoint sets among $S_1, \ldots, S_n$. The reduction is as follows. We are given a graph $G = (V, E)$. Assume without loss of generality that $V = \{1, \ldots, n\}$ and that the edges in E are indexed from 1 to m. The reduction outputs the number $m = |E|$ and the $n = |V|$ sets $S_1, \ldots, S_n$ where each $S_i \subseteq \{1, \ldots, m\}$ is the set of edges that touch vertex i.

   This reduction can clearly be implemented in time polynomial in the input size, i.e., in time $\mathrm{poly}(n)$: for each $i = 1, \ldots, n$ output the list of edges that touch vertex i by checking all m edges.

   We now prove correctness. We start with (a). Assume G has an independent set I of size $\beta|V| = \beta n$. Then the sets S_i for all $i \in I$ are pairwise disjoint since if $S_i \cap S_j \neq \emptyset$ for some $i \neq j$ then by definition there is an edge in G that touches both i and j, which means that $\{i, j\} \in E$, but this is impossible since I is an independent set. We now prove (b). Assume that there are more than αn pairwise disjoint sets among $S_1, \ldots, S_n$, and let $I \subseteq \{1, \ldots, n\}$ be their indices. Then we claim that I is an independent set. Indeed, let $\{i, j\} \in E$ be any edge in G. Then, by our construction, $S_i \cap S_j \neq \emptyset$ (since the edge $\{i, j\}$ touches both i and j and is therefore contained in both S_i and S_j) and therefore either i or j are not in I.

   (b) Our goal now is to show a reduction in the reverse direction: given a number m and sets $S_1, \ldots, S_n \subseteq \{1, \ldots, m\}$, the reduction needs to output a graph $G = (V, E)$ such that: (a) if there are βn pairwise disjoint subsets among $S_1, \ldots, S_n$ then G has an independent set of size $\beta|V|$, and (b) if there are at most αn pairwise disjoint sets among $S_1, \ldots, S_n$ then the largest independent set in G is of size at most $\alpha|V|$. The reduction is as follows. We are given a number m and sets $S_1, \ldots, S_n \subseteq \{1, \ldots, m\}$. The reduction outputs the graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ and E given by

$$E = \{\{i, j\} \ : \ i \neq j \ \wedge \ S_i \cap S_j \neq \emptyset\}.$$

   In other words, we connect by an edge any two vertices that correspond to sets that are not disjoint.

   This reduction can clearly be implemented in time polynomial in the input size, i.e., in time $\mathrm{poly}(n, \log m, \sum_i |S_i|)$ (notice that m might be exponential in the input size): for each pair $\{i, j\}$ we check if their sets intersect, and decide whether to put an edge between them accordingly.

   We now prove correctness. We start with (a). Assume there are βn pairwise disjoint subsets among $S_1, \ldots, S_n$, and let I denote their indices. Then clearly I is an independent set in the graph we constructed since we only put edges between sets that intersect. We now prove (b). Assume that G has an independent set I of size more than $\alpha|V|$. Then by our construction, the sets S_i for $i \in I$ are pairwise disjoint.

4. (a) The optimal solution must place item i in one of the bags, and the weight of that bag is therefore at least a_i.

(b) Consider the total weight of bags in the optimal solution. On one hand, it is clearly equal to the total sum of weights of items, $\sum_{i=1}^{n} a_i$. On the other hand, since no bag is heavier than $|\mathsf{OPT}|$, it is at most $m|\mathsf{OPT}|$. Combining these two observations leads to the required inequality.

(c) Let j be the heaviest bag in ALG, let i be the last item placed into j, and let w be the weight of j just before i was placed into it. Notice that no bag in ALG can have weight smaller than w since otherwise item i would not have gone into bag j. This implies that the sum of weights of all items satisfies $\sum_{i=1}^{n} a_i \geq mw$. By item (b) we see that $w \leq |\mathsf{OPT}|$, as required.

(d) As before, let j be the heaviest bag in ALG, let i be the last item placed into j, and let w be the weight of j just before i was placed into it. Notice that the solution found by the algorithm has value $|\mathsf{ALG}| = w + a_i$. By item (c) we have that $w \leq |\mathsf{OPT}|$. By item (a) we know that $a_i \leq |\mathsf{OPT}|$. Therefore, $|\mathsf{ALG}| \leq 2|\mathsf{OPT}|$, as required.