Mid Semester Examination, Autumn 2003-04

Solutions

---

1. FALSE. We additionally require the condition $B \in \mathrm{NP}$.

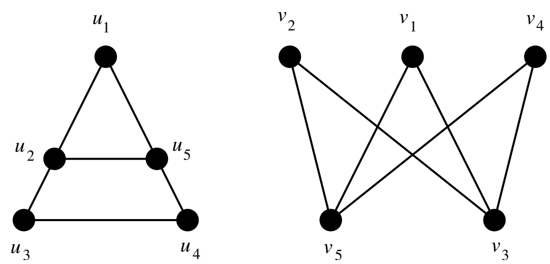
2. FALSE. Let α be an input of size n for A. Call the reduction map f, i.e., $f(\alpha)$ is an input for B. Since f is computable in time $O(n^k)$, the string $f(\alpha)$ can be of length as big as $O(n^k)$. Subsequent application of the algorithm for B then runs in $O((n^k)^k)$, i.e., $O(n^{k^2})$, time. For $k > 1$ we have $k^2 > k$.

3. TRUE. Description of a cycle in G of length $\geqslant \lfloor n(G)/2 \rfloor$ is a succinct certificate for $\langle G \rangle$ to be in BIGCYCLE, i.e., BIGCYCLE $\in$ NP. For the NP-hardness, I show HAMCYCLE $\leqslant_P$ BIGCYCLE. Let G be an instance for HAMCYCLE with m vertices. Add (exactly) m isolated vertices to G, thereby obtaining a graph G' on $2m$ vertices. It is evident that G' has a cycle of length m (i.e., $\geqslant \lfloor n(G')/2 \rfloor$), if and only if G has a Hamiltonian cycle.

4. FALSE (under the assumption that NP $\neq$ coNP). I first show that SMALLCYCLE $\in$ coNP. If $\langle G \rangle$ is not in SMALLCYCLE, then we can convince one about this fact either by indicating that G is acyclic or by explicitly providing a cycle in G of length $> \lfloor n(G)/2 \rfloor$. That's a succinct and poly-time verifiable disqualification for G. Now if SMALLCYCLE were NP-complete, we would have NP $=$ coNP.

5. FALSE. An NTM accepts, if and only if there is an accepting branch of computation. For the given algorithm a branch accepts, if and only if a cycle $(u_1, \ldots, u_m)$ is detected and some choice of $v_1, \ldots, v_k$ does not lead to a cycle. However, another choice of $v_1, \ldots, v_k$ may lead to a big cycle. That's not taken care of.

6. FALSE. The following two graphs are not isomorphic, since the left graph contains a triangle, whereas the right one, being bipartite, is triangle-free. But the given algorithm accepts this pair with successive choices of u and v as shown (with subscripts indicating the iteration number). (Note that throughout the computation on this example, the (sorted) degree sequences of the two graphs remain the same.)



7. FALSE. Since NL $=$ coNL, the concepts NL-complete and coNL-complete are the same. We know that PATH is NL-complete and so must be $\overline{\text{PATH}}$. However, PATH $\cap$ $\overline{\text{PATH}}$ is the empty language, which can not be complete in any useful complexity class.

8. FALSE. Since it is widely assumed that NP $\neq$ coNP, the reasoning of the previous exercise does not work. I will anyway use a kind of intersection, but not at the level of languages. I first show that the languages

$$\begin{aligned} \text{BIGCLIQUE} &:= \{\langle G \rangle \mid G \text{ has a clique of size} \geqslant 1 + n(G)/2\} \text{ and} \\ \text{BIGINDSET} &:= \{\langle G \rangle \mid G \text{ has an independent set of size} \geqslant 1 + n(G)/2\} \end{aligned}$$

on undirected graphs are both NP-complete. BIGCLIQUE is clearly in NP — a listing of the vertices in a big clique constitutes a succinct certificate. One can reduce CLIQUE to BIGCLIQUE in poly time as follows. Let $\langle G, k \rangle$ be an instance for CLIQUE. We want to produce a graph G' such that G' has a big clique if and only if G has a k-clique. Call $m := n(G)$. If $k \geqslant 1 + m/2$, then G' is obtained from G by adding $2k - m - 2$ isolated vertices to G. On the other hand, if $k < 1 + m/2$, add $m - 2k + 2$ new vertices to G and edges connecting each pair of these new vertices and each new vertex to each old vertex of G. It reduction from BIGCLIQUE to BIGINDSET that maps a graph G to its complement $\bar{G}$.

I claim that $\mathrm{BIGCLIQUE} \cap \mathrm{BIGINDSET} = \emptyset$. Suppose not, i.e., some $\langle G \rangle$ belongs to this intersection. Let S be a big clique and T a big independent set in G. If u, v are two distinct vertices in $S \cap T$, then the edge (u, v) is both in G (a part of a clique) and not in G (a part of an independent set). Thus $|S \cap T| \leqslant 1$. But then $n(G) \geqslant |S \cup T| = |S| + |T| - |S \cap T| \geqslant (1 + n(G)/2) + (1 + n(G)/2) - 1 = 1 + n(G) > n(G)$, which is absurd.

9. TRUE. Here is a deterministic log-space algorithm for TRIANGLE-FREE:

1. for each triple (u, v, w) of vertices in G
2. if all of (u, v), (v, w) and (w, u) are edges of G, *reject*.
3. *Accept*.

This algorithm need only store the three vertices u, v, w and must employ a mechanism to step through all possibilities – both achievable in log-space.

10. TRUE. We already know that 3COLOR is NP-complete. Because we have used the same reduction mechanism (poly-time) for defining completeness in both NP and PSPACE, PSPACE $=$ NP implies that 3COLOR is PSPACE-complete. For proving the converse, assume that 3COLOR is PSPACE-complete. Take any $L \in$ PSPACE. By definition we then have $L \leqslant_P$ 3COLOR. But then this reduction followed by an NP algorithm for 3COLOR solves L in nondeterministic poly-time, implying that $L \in$ NP, i.e., PSPACE $\subseteq$ NP. The reverse inclusion is well-known.