

Solution Set 4

Posted: February 22

1. (a) False. By the Time Hierarchy Theorem there are problems that are solvable in time $O(2^{3(2n)^{100}})$ that are not in $\text{TIME}(2^{n^{100}})$. Nevertheless, a problem with an algorithm running in time $O(2^{3(2n)^{100}})$ is in EXP .
 - (b) True. If both $P = NP$ and $NP = \text{EXP}$, then $P = \text{EXP}$. But we know that $P \neq \text{EXP}$, by the Time Hierarchy Theorem.
 - (c) False. Reductions may have running time $O(n^{k'})$ for $k' > k$, which spoils the straightforward argument. Formally the proof that it is false goes as follows. The premise implies $P = NP$. But using the Time Hierarchy Theorem exactly as we did in (a), there are languages in P that do not have algorithms running in time $O(n^k)$.
 - (d) False. The empty language \emptyset and its complement cannot be NP-complete, because no other language can reduce to them. To see this, suppose we have a reduction f from $A \in NP$ to \emptyset (where $A \neq \emptyset$). Pick an arbitrary $w \in A$. Since f is a reduction, we must have $f(w) \in \emptyset$, a contradiction.
(Note, however, that if $P = NP$, then every language other than \emptyset and $\bar{\emptyset}$ is NP-complete. Take an arbitrary language $L \in NP$, and pick some $w \in L$, and some $w' \notin L$. Here's a reduction from an arbitrary language $A \in NP$ to L : given an instance x , our reduction f decides in polynomial time whether $x \in A$ or not, and in the first case it outputs w , in the second case it outputs w' .)
 - (e) True. By the definition of "complete." Every language in NP is in EXP since $NP \subseteq \text{EXP}$, and thus it reduces to every EXP -complete language.
2. We will reduce 2-COLORABLE to 2-SAT, which we showed to be in P . Given a graph G , our reduction produces the following set of clauses: label the vertices of G with variables x_1, x_2, \dots, x_n ; for every edge between vertices labelled x_i and x_j , produce the clauses $(x_i \vee x_j)$ and $(\bar{x}_i \vee \bar{x}_j)$.

Clearly this reduction runs in polynomial time. We now argue that "yes maps to yes." If G is 2-colorable, then pick one of the two colors in a 2-coloring of G and assign the associated variables TRUE. Every one of the clauses is satisfied, because the only way to fail to satisfy a clause is if the two endpoints of some edges were the same color.

We argue that "no maps to no." Suppose we have a satisfying assignment to the set of clauses produced by the reduction. Then the two variables associated with a given edge must have different truth assignments. Therefore, if we color the vertices of G associated with TRUE variables red, and the other vertices of G green, we will have produced a valid 2-coloring of G , and hence G is 2-colorable.

3. The key here is that H is fixed. Suppose it has k vertices. Given an input G , we can simply exhaustively try all potential subgraph isomorphisms. Specifically, we try every subset of k vertices of G , and for each subset we try every one of the $k!$ possible ways of mapping it to the vertices of H . We can easily check, whether H equals the subgraph induced by a given subset of G 's vertices (permuted according to a given mapping) – in fact this can be done in time polynomial in k .

Thus the overall running time of the algorithm is

$$\binom{|G|}{|H|} \cdot k! \cdot k^{O(1)}$$

Note that $\binom{n}{k} \leq n^k$, and $k! \leq k^k$, so we have an overall running time of $|G|^{O(k)}$. Since k is fixed, this is polynomial in the size of the input, G .

4. Following the hint, we will build up a table T which has a TRUE/FALSE entry for each $0 \leq B' \leq B$ telling us whether some multiset of the x_i sum to exactly B' . Specifically, let T be an array with $B + 1$ entries. Initialize $T[i]$ to be FALSE for all i . For $B' = 0, 1, 2, \dots, B$ do:
- if some $x_i = B'$ then $T[B'] = \text{TRUE}$.
 - else if for some x_i we have $T[B' - x_i] = \text{TRUE}$, then set $T[B'] = \text{TRUE}$ (otherwise leave $T[B']$ set to FALSE).

At the end, we accept iff $T[B]$ is TRUE.

In each step we need to check all of the x_i , so the running time for one iteration of the loop is $O(n)$. The overall running time is $O(Bn)$ which is polynomial in the size of the input (here we use crucially that B is presented in unary).

We should also argue that this algorithm is correct. We claim that after the B' -th iteration, all entries of T up to and including B' are correct (i.e. they are TRUE if there is a multiset of the x_i summing to that value, and FALSE otherwise). The base case (when $B' = 0$) is clearly satisfied. Now assume T is correct up to and including $B' - 1$. There is a non-empty multiset summing to exactly B' iff for some i , there is a non-empty multiset summing to $B' - x_i$. By induction $T[B' - x_i]$ is correct, and so we correctly fill in $T[B']$ in the B' -th iteration of the main loop.