Discussion Solutions 8

Problem One: NP

i. Given a sequence of numbers $x_1, x_2, \ldots, x_n$, an ***ascending subsequence*** is a subsequence of the original sequence (that is, some elements of the sequence taken in the original order in which they appear) such that each term is larger than the previous term. For example, given the sequence 2, 3, 0, 1, 4, the subsequence **2**, **3**, **4** is an ascending subsequence, as is **0**, **1**, **4**. Let $ASCEND = \{ \langle x_1, x_2, \ldots, x_n, k \rangle \mid$ There is an ascending subsequence of $x_1 \ldots x_n$ with length at least k. $\}$ Prove that $ASCEND \in \mathbf{NP}$ by designing a polynomial-time verifier for it.

One possible verifier is as follows:

> V = "On input $\langle x_1, x_2, \ldots, x_n, k, v_1, v_2, \ldots, v_k \rangle$:
>
> Check that $v_1 \ldots v_k$ is a subsequence of $x_1 \ldots x_k$.
>
> Check that $v_1 < v_2 < \ldots < v_k$.
>
> If so, accept; otherwise, reject."

First, we prove that V is a verifier for $ASCEND$ by showing that $\langle x_1, x_2, \ldots, x_n, k \rangle \in ASCEND$ iff there is some $v_1, v_2, \ldots, v_k$ such that V accepts $\langle x_1, x_2, \ldots, x_n, k, v_1, v_2, \ldots, v_k \rangle$. To see this, note that $\langle x_1, x_2, \ldots, x_n, k \rangle \in ASCEND$ iff there exists some $v_1, \ldots, v_k$ such that $v_1, \ldots, v_k$ is an ascending sequence in $x_1, x_2, \ldots, x_n$ of length k. Finally, note by construction that V accepts $\langle x_1, \ldots, x_n, k, v_1, \ldots, v_k \rangle$ iff $v_1, \ldots, v_k$ is an ascending subsequence of $x_1, \ldots, x_n$ of length at least k. Thus there is some $v_1, v_2, \ldots, v_k$ such that V accepts $\langle x_1, x_2, \ldots, x_n, k, v_1, v_2, \ldots, v_k \rangle$ iff $\langle x_1, x_2, \ldots, x_n, k \rangle \in ASCEND$.

To see that this can be done in polynomial time, note that we can check whether $v_1 \ldots v_k$ is a subsequence of $x_1 \ldots x_k$ by maintaining two indices i and j into x and v, respectively. At each point, we check whether $x_i = v_j$. If so, we increment i and j, and otherwise just increment i. If at the end we find that $j = k$, we know that $v_1 \ldots v_k$ is a subsequence of $x_1 \ldots x_n$; otherwise it is not. This requires a total of $O(n)$ iterations, each of which takes polynomial time, so we can check if $v_1 \ldots v_k$ is a subsequence of $x_1 \ldots x_n$ in polynomial time. We can check whether the sequence $v_1 \ldots v_k$ is ascending by checking $O(n)$ adjacent pairs of terms, each in polynomial time. Thus the total runtime of this verifier is a polynomial.

ii. In an undirected graph $G = (V, E)$, a ***dominating set*** is a set $D \subseteq V$ such that every node in V either belongs to D or is connected to a node in D by an edge. Every graph has a dominating set consisting of every node in the graph, though it's unclear whether smaller dominating sets exist.

Let $DS = \{ \langle G, k \rangle \mid G$ is an undirected graph containing a dominating set with at most k nodes $\}$. Prove that $DS \in$ **NP** by designing a polynomial-time verifier for it. (It turns out that $DS \in$ **NPC** as well, though that proof is a bit harder.)

One possible verifier is

$X =$ "On input $\langle G, k, D \rangle$:

Check that $|D| \leq k$; if not, reject.

Check that every element of D is a node in G; if not, reject.

Remove all nodes in D from G.

Remove from G all nodes connected by an edge to a node in D.

If no nodes remain in G, accept; otherwise reject."

First, we prove that this is a valid verifier for DS. To do this, we will prove that $\langle G, k \rangle \in DS$ iff there is some D such that X accepts $\langle G, k, D \rangle$. Note that $\langle G, k \rangle \in DS$ iff there is some dominating set in G containing at most k nodes. Finally, note that by construction, X accepts $\langle G, k, D \rangle$ iff D is a dominating set in G with at most k nodes. Thus there is some D such that X accepts $\langle G, k, D \rangle$ iff $\langle G, k \rangle \in DS$, so X is a verifier for DS.

To see that X runs in polynomial time, note that we can verify whether the size of D is at most k in polynomial time by counting up the nodes in D in polynomial time each and repeating this at most $O(|V|)$ times. We can check that each node in D is a node in G in polynomial time by iterating across the $O(|V|)$ nodes in D and comparing them to the $O(|V|)$ nodes in G; this requires at most $O(|V|^2)$ comparisons, each of which takes polynomial time. We can remove all the nodes in D from G using a similar approach, so this step takes polynomial time. We can remove all the nodes in G that are connected by an edge to a node in D by iterating over all $O(|E|)$ edges in the graph and testing whether each endpoint belongs to D by making $O(|V|^2)$ comparisons using this same approach. Finally, checking whether G has no more nodes could be done in $O(1)$ time. Since every step takes polynomial time, the overall runtime of this verifier is a polynomial.

Problem Two: NP-Completeness

i. Prove that *SETPACK* ∈ **NP** by designing an NTM that decides it in polynomial time.

Here is one polynomial-time NTM for *SETPACK*:

> *M* = "On input ⟨S_1, S_2, …, S_n, k⟩:
>
> Nondeterministically guess k different sets from S_1 … S_n.
>
> For each of the k sets chosen this way:
>
> Add each element of the set to a list.
>
> If a duplicate is discovered, reject.
>
> Otherwise, accept."

To see that this NTM runs in polynomial time, note that we can guess k different sets in polynomial time by iterating over the n sets we are given as input and nondeterministically choosing, for each set, whether to pick that set or not pick that set. Every time a set is chosen to be included, we can increment a counter and then stop as soon as the counter reaches k. Overall, this takes polynomial time.

Having then guessed the sets to pick, the NTM then takes polynomial time to iterate over each set and add its elements to the list. We can see this as follows: the total number of elements in the sets is some polynomial in the length of the input string, and each step of the verification requires at most quadratically many comparisons between elements. Overall, this requires only polynomial time to complete.

Finally, we show that this NTM is correct by showing that there is some choice it can make such that it accepts ⟨S_1, S_2, …, S_n, k⟩ iff ⟨S_1, S_2, …, S_n, k⟩ ∈ *SETPACK*. To see this, note that there is some choice that the NTM can make such that it accepts ⟨S_1, S_2, …, S_n, k⟩ iff there is some collection of k sets chosen from S_1, S_2, …, S_n such that those sets do not have any elements in common. This occurs iff ⟨S_1, S_2, …, S_n, k⟩ ∈ *SETPACK*, as required. ■

ii. Using this reduction, prove that *SETPACK* ∈ **NPC** by showing *INDSET* $\leq_P$ *SETPACK*.

Proof: We know from (i) that *SETPACK* ∈ **NP**, so we need to show that it is **NP**-hard. To do so, we prove that the reduction is valid (that is, ⟨*G*, *k*⟩ ∈ INDSET iff ⟨S_1, …, S_n, *k*⟩ ∈ *SET-PACK*) and that the reduction can be computed in polynomial time.

To see that this reduction can be computed in polynomial time, note that each set created has size at most |*E*| and that there are |*V*| different sets. Thus the total number of elements generated in the sets is at most O(|*E*| + |*V*|), which is a polynomial in the size of the input. Constructing each set might require polynomial time to scan over all the edges in the graph, so collectively we spend polynomial time constructing all the sets.

To show this is a valid reduction, we will prove that ⟨*G*, *k*⟩ ∈ INDSET iff ⟨S_1, S_2, …, S_n, *k*⟩ ∈ SETPACK. First, suppose that ⟨*G*, *k*⟩ ∈ INDSET. Then there is an independent set $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ in G. We prove that there is packing of k of the sets S_1 … S_k, so ⟨S_1, S_2, …, S_n, *k*⟩ ∈ SETPACK. To see this, choose the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$. To show that this is a legal packing, assume for the sake of contradiction that it is not and that some edge e is in two of the sets; call those sets S_i and S_j. Since e ∈ S_i, one of the endpoints of e must be v_i, and since e ∈ S_j, one of the endpoints of e must be v_j. Thus e = {v_i, v_j}. But since we picked S_i and S_j in our set packing, this means that v_i and v_j are in the independent set, but that e is an edge connecting them, a contradiction. Thus our assumption was wrong and $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ is a valid set packing.

To prove the other direction of implication, we show that if $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ is a valid set packing, then $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ is an independent set in G. To see this, suppose for the sake of contradiction that it is not, and that there is some edge e = {v_i, v_j} connecting two of the nodes in this independent set. Since e is incident to both v_i and v_j, this means that e ∈ S_i and e ∈ S_j, which contradicts the fact that $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ is a set covering. Thus our assumption was wrong and $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ is an independent set in G. ■