

7.1 Answer TRUE or FALSE for each of the following parts.

- (a) $2n = O(n)$. TRUE
- (b) $n^2 = O(n)$. FALSE
- (c) $n^2 = O(n \log^2 n)$. FALSE
- (d) $n \log n = O(n^2)$. TRUE
- (e) $3^n = O(2^n)$. FALSE. $3^n = 2^{n \log_2 3} = 2^{O(n)}$.
- (f) $2^{2^n} = O(2^{2^n})$. TRUE

7.7 Show that NP is closed under union and concatenation.

First, I will show that NP is closed under union. That is, given $L_1 \in NP$, and $L_2 \in NP$, then $L_1 \cup L_2 \in NP$. First, since $L_1 \in NP$, then there exists a TM M_1 that such that given a string w and a certificate c , where $|c| = poly(w)$, then M can verify in polynomial time whether w is an element of L_1 . Likewise for L_2 , there exists a TM M_2 such that given a string w and a certificate c , where $|c| = poly(w)$, then M can verify in polynomial time whether w is an element of L_2 . Now, I can construct N , a TM for $L_1 \cup L_2$ that on given a certificate c' and input w' can verify in polynomial time whether w' is an element of $L_1 \cup L_2$ as follows:

Let the new certificate c' consist of two parts: the first part is a bit to indicate the language to which w' belongs, and the second part is the certificate for that w' in that language. Then, on input c', w' , N reads c' and based on c' runs either M_1 or M_2 , feeding that machine w' and the certificate for w' in that language. c' is clearly polynomial in the length of this inputs, as it is only a constant number of bits longer than the original certificate for w .

To show that NP is closed under concatenation, I must show that given $L_1 \in NP$, and $L_2 \in NP$, then $L_1 \cdot L_2 \in NP$. In other words, given a string $w \in L_1$, and a string $q \in L_2$, then a string $w \cdot q \in L_1 \cdot L_2$. Assume L_1 and L_2 are defined as above, and have corresponding TMs M_1 and M_2 . I create a new TM N or $L_1 \cdot L_2$ that on given a certificate c' and input w' can verify in polynomial time whether w' is an element of $L_1 \cdot L_2$ as follows:

Let the certificate c' consist of two parts: the first part is the bit position of the break between the string belonging to L_1 and the string belonging to L_2 , and the second part consists of two certificates for the languages to which the string belongs. Since c' is longer than the original certificates by at most $\log |w_1 + w_2|$ bits, (which indicate the position of the break), then $|c'|$ is polynomial in the size of the inputs.

7.9 Let $CONNECTED = \{ \langle G \rangle \mid G \text{ is a connected undirected graph} \}$. Analyze the algorithm given on page 145 to show that this language is in P.

The algorithm given in Sipser, p. 146 is :

$M =$ “ On input $\langle G \rangle$, the encoding of a graph G :

- (a) Select the first node of G and mark it.
- (b) Repeat the following stage until no new nodes are marked:
 - i. For each node in G , mark it if it is attached by an edge to a node that is already marked.
- (c) Scan all nodes of G to determine whether they all are marked. If they are, *accept*; otherwise, *reject*.”

Step 1 takes $O(1)$ time. To analyze Step 2, note that for a given node, checking all edges can take at most $O(n^2)$ time (as at most, there are $O(n^2)$ nodes.) Since there are n nodes, marking once through for each node takes $O(n^3)$ time. Further, in the worst case, since only one new node is marked on each pass, we may have to repeat passing through all of the nodes n times. In total, step 2 has a running time of $O(n^4)$. Step 3 takes $O(1)$ time. Hence, the algorithm has a running time of $O(n^4)$, and therefore $CONNECTED \in P$.

- 7.11 Call graphs G and H isomorphic if the nodes of G may be reordered so that it is identical to H . Let $ISO = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are isomorphic graphs} \}$. Show that $ISO \in NP$.

To show that $ISO \in NP$, we simply show how given an input w and a certificate c for w , a verifier for ISO would run in polynomial time. The certificate c is the mapping from G to H that demonstrates G and H are isomorphic. Specifically, the certificate can be specified as pairs of nodes (g, h) , where $g \in G$ and $h \in H$. An isomorphism is a bijective mapping: (surjective and injective, or 1-to-1 and onto), so there can only be one node in H to which a node in G is mapped, and every node in G is mapped once. Therefore, there are $|G| = |H| = n$ elements in the mapping, and each can be specified with $O(\log n)$ bits. Therefore, the mapping is polynomial in the size of the input. Now, we need to show that the verifier runs in poly time. The input to the verifier are the descriptions of G and H : each graph description consists of a list of nodes, and a list of edges.

- (a) For each node $g \in G$,
 - i. Use the certificate to determine that mapping f to a node $h \in H$. That is, check $f(g) = h$, and that the node in H is previously unmarked. Mark that node. (If a node is previously marked, reject). Assuming the previous conditions are met,
 - ii. run through list of edges, checking for edges of the form (g, x) , where x is any other node in G . For each edge that is found, compute the mapping $f(x)$, and check that if (g, x) is an edge in G , then $(f(g), f(x))$ is an edge in H . Mark each node and edge for which the mapping is found. If an edge is found that has already been marked, reject.
- (b) Check that all nodes and edges in H are marked.

This algorithm runs in polynomial time. Given a node g , checking the mapping and marking the node $f(g)$ takes linear time. For each node, running through the list of edges takes linear time w.r.t. the input, as the edges are listed in the description. For each edge, checking the mapping takes linear time. So, for each node, computing step two takes $O(n^2)$ time, and as there are n nodes, this takes a total of $O(n^3)$ time. Checking all nodes and edges are marked is linear time. Hence, the verifier can verify the input string in polynomial time. Hence, $ISO \in NP$.

- Last Let G be a directed graph of n vertices. How many bits do you need to describe G ? Also, how many bits (again as a function of n) do you need to describe a permutation of the vertices of G ? (Your answer should use the big-O notation; but not $O(n^4)$ if you can make it $O(n^3)$.)

$G = (V, E)$ is a directed graph of n vertices and some number of edges. The graph description can be a list of vertices and a list of edges. The vertices can be named by numbers 0 through $n - 1$. Each vertex name can be written with $O(\log n)$ bits; the total number of bits for all vertices is $O(n \log n)$. The edges can be listed as pairs of vertices. Since the edges are directed, the total number of edges possible is $\binom{2 \cdot n}{2}$. So, the number of edges is $O(n^2)$. The number of bits to specify each edge is the number of bits to specify vertex 1 + the number of bits to specify vertex 2 + $c = O(\log n)$. Therefore, the total number of bits to specify the graph is $O(n \log n) + O(n^2 \log n) = O(n^2 \log n)$. There are many ways to describe a permutation of the vertices. Given n vertices, there are $n!$ possible permutations of those vertices. You can specify which permutation simply by enumerating all possible permutations in lexicographic order, and then labelling them by their number in that ordering. The number of bits to specify a permutation is then $\log n!$ and by Sterling's approximation, $\log n! = O(n \log n)$.