Introduction to the Theory of Computation
Homework #1 Solutions

1. (Exercise 0.4) The *Cartesian product* of two sets A and B, written $A \times B$, is the set of ordered pairs (a, b) where $a \in A$ and $b \in B$. How many elements does $A \times B$ have? Write $|A \times B|$ in terms of $|A|$ and $|B|$.

Solution: $|A \times B| = |A||B|$, since we can choose any element of A and then independently choose any element of B.

2. Given two sets A and B, let F be the set of *functions* $f : A \to B$. In other words, each $f \in F$ assigns an element of B to each element of A. What is $|F|$ in terms of $|A|$ and $|B|$?

Solution: Each function from A to B is a string of length $|A|$ over the alphabet B. Each such string consists of $|A|$ independent choices, each of which has $|B|$ possibilities. The total number of such strings is

$$\underbrace{|B| \times |B| \times \cdots \times |B|}_{|A| \text{ times}} = |B|^{|A|} \ .$$

3. For a set A, the *power set* $\mathcal{P}(A)$ is the set of all subsets of A. Fill in the blank: we can think $\mathcal{P}(A)$ as the set of functions from A into $\langle$blank$\rangle$.

Solution: Specifying a subset $S \subseteq A$ is equivalent to giving you a function $F : A \to \{0, 1\}$ such that $F(a) = 1$ if $a \in S$ and 0 if $a \notin S$. Thus $B = \{0, 1\}$. Of course, any 2-element set will do. (We can also see that $|B|$ should be 2 given the answer to the previous question, since $|\mathcal{P}(A)| = 2^{|A|}$.)

4. A function $f : A \to B$ is *one-to-one* if for all $a_1 \neq a_2$, $f(a_1) \neq f(a_2)$ — or, to put it differently, $f(a_2) = f(a_2)$ implies $a_1 = a_2$. It is *onto* if for all $b \in B$, there exists an $a \in A$ (not necessarily unique) such that $f(a) = b$.

Consider the following functions from $\mathbb{N}$ to $\mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ is the set of natural numbers (non-negative integers). For each function, state whether it is onto, and whether it is one-to-one. (Note that 2. and 3. can be easily expressed in terms of x's binary digit sequence.)

1. $f(x) = x + 1$

2. $f(x) = \begin{cases} x + 1 & \text{if } x \text{ is even} \\ x - 1 & \text{if } x \text{ is odd} \end{cases}$

3. $f(x) = x/2$ (rounded down)

4. $f(x) = (x-4)^2$

Solution: 1. is 1-1 but not onto; 2. is 1-1 and onto; 3. is onto but not 1-1; and 4. is neither 1-1 nor onto.

5. (Exercise 0.7) A binary relation $\sim$ is *reflexive* if $x \sim x$ for all x, *symmetric* if $x \sim y$ if and only if $y \sim x$, and *transitive* if $x \sim y$ and $y \sim z$ implies $x \sim z$.

Give examples of relations (from math, life, or anywhere else) which are

1. Reflexive and transitive but not symmetric

2. Transitive and symmetric but not reflexive

3. Symmetric and reflexive but not transitive

Solution (for instance): for 1. let x and y be people and say that $x \sim y$ if $x = y$ or if x is an ancestor of y. More generally, let x and y be vertices in a directed graph and say that $x \sim y$ if there is a path of length 0 or more from x to y. This is transitive since your ancestor's ancestor is your ancestor, or since we can link paths $x \to y$ and $y \to z$ to make a path $x \to z$.

For 2. let x and y be people and say $x \sim y$ if they are full brothers (i.e. sharing both parents) but are not the same person. Or two people living in the same country, etc. This would be an equivalence relation except that we forbid $x \sim x$.

For 3. let x and y be people and say $x \sim y$ if they live within 1 mile of each other (including if they are the same person). More generally, let $x \sim y$ be vertices in an undirected graph and say $x \sim y$ if they are neighbors; this is not transitive since your neighbor's neighbor might not be your neighbor. (Another cute example is brothers, since your half-brother's half-brother might not share either parent with you — or in-laws.)

6. (Exercise 0.11) Find the error in the following proof that all horses are the same color.

Definition: A set of horses is *uniform* if all the horses in it are of the same color.

Lemma: if U, V, and $U \cap V$ are uniform, then so is $U \cup V$. Proof: obvious.

Base case: a set of one horse is clearly uniform.

Induction step: suppose all sets of less than k horses are uniform. We wish to show that any set of k horses is also. Let S be a set of k horses where $k > 1$. Then S is the union of two sets U, V of size $k-1$ (for instance, let U be S minus the first horse, and V be S minus the second horse). Since U and V are smaller than S, they are uniform by the inductive hypothesis, and since their intersection is also smaller than S, it is too. By the Lemma above, $U \cup V = S$ is also uniform. QED.

Solution: The problem is with the induction step, and specifically with the Lemma (beware of proofs that say "obvious"!) The null set is uniform (after all, it doesn't contain two horses of different colors) but if U and V are disjoint the Lemma is false. The problem arises when $k = 2$: we then have two sets, each of size $k - 1 = 1$, but they can both be uniform without their unions being uniform since their intersection is empty.

7. Let $\sum(S)$ be the sum of the elements in a set, and $\prod(S)$ be their product. If S and T are disjoint, what is $\sum(S \cup T)$ and $\prod(S \cup T)$? Conclude from this what $\sum(\emptyset)$ and $\prod(\emptyset)$ should be ($\emptyset$ is the empty set).

Solution: if S and T are disjoint, clearly $\sum(S \cup T) = \sum(S) + \sum(T)$ and $\prod(S \cup T) = \prod(S) \cdot \prod(T)$. (By the way, what if they are not disjoint?)

Now, since $S \cup \emptyset = S$ for all S, we have $\sum(S) + \sum(\emptyset) = \sum(S)$ and $\prod(S) \cdot \prod(\emptyset) = \prod(S)$. It follows that $\sum(S) = 0$ and $\prod(S) = 1$ — in other words, the sum of nothing is zero, but the product of nothing is one!

8. True or false: every element of the empty is set is purple.

Solution: Of course! It contains no non-purple elements. Consider also the following: let P be the set of purple things and G be the set of green things. Then every element of $P \cap G$ is both purple and green. But (assuming for simplicity that each thing is only one color) $P \cap G = \emptyset$.

Note also that every element of the empty set is named Bob.

9. Suppose I have n cities, and the distances between them are given as n-bit numbers. How many bits could the length of the shortest tour (or, for that matter, the longest tour) possibly have?

Now, suppose I use the yes/no oracle for the decision version of the Travelling Salesperson Problem to find the length of the shortest tour. Show that if I do this with a "linear scan" (starting at 0 and searching upward) it might take exponential time (i.e. an exponential number of calls to the oracle), but if I do a binary search it will only take polynomial time. Remember that when I say "exponential" or "polynomial", I mean as a function of n.

Solution: since a tour has length n, the longest the tour could be is n times the maximum length. Since the lengths have n bits the maximum length is 2^n, so the longest a tour could be is $L_{\max} = 2^n \times n$. This number has $\log_2 L_{\max} = \log_2(2^n n) = n + \log_2 n = O(n)$ bits, only slightly more than the number of bits each distance has.

Now, if we start out at 0 and search upward towards $L_{\max}$ linearly, it might take as many as $L_{\max} = 2^n n$ steps, which is exponential in n. However, a binary search that starts at $L_{\max}/2$ and then goes to $L_{\max}/4$ or $3L_{\max}/4$, and so on, will only take $\log_2 L_{\max} = O(n)$ steps, which is polynomial in n.

10. Now that I know the length of the shortest tour, how do I find the shortest tour in a polynomial number of calls to the oracle? (Hint: I can modify the problem, i.e. the set of cities, the distances between them, and the desired distance, in any manner I desire.)

Solution: for each edge, increase its length by 1 and use the oracle to recalculate the length of the new shortest tour. The edges used by the shortest tour are those where increasing their length increases the length of the tour. (We need to be a little careful if there are multiple tours of the same length — do you see how to handle this case?)

Since a graph with n vertices has at most $\binom{n}{2} = O(n^2)$ edges, we just have to repeat our procedure for finding the length of the shortest tour a polynomial number of times. Since each time takes a polynomial number of calls to the oracle, and since the product of a polynomial is a polynomial, the total number of calls to the oracle is still a polynomial (in fact $O(n^3)$).