Homework 4

Due: March 1, 2023

All homeworks are due at 11:59 PM on Gradescope.

Please do not include any identifying information about yourself in the handin, including your Banner ID.

Be sure to fully explain your reasoning and show all work for full credit.

Problem 1

Recall that an *equivalence relation* is one that is reflexive, symmetric, and transitive.

Determine whether or not each of the following relations is an equivalence relation. Be sure to justify your answers.[1] In particular: if a relation is *not* an equivalence relation, which of the above properties does it not satisfy?

a. The relation R on $\mathbb{Z}$ defined by the set of ordered pairs of integers:

$$R = \left\{(a, b) \;\middle|\; |a - b| \geq 3\right\}.$$

Solution:

This relation is not reflexive: for example, $(1, 1)$ is not in the relation, since $|1 - 1|$ is not greater than or equal to 3. (In fact, it is irreflexive.)

This relation is not transitive. We have $(1, 6), (6, 2) \in R$, as both $|1 - 6|$ and $|6 - 2|$ are greater than or equal to 3. If it were transitive, $(1, 2)$ would be in the relation. However, $|1 - 2|$ is not greater than or equal to 3, and therefore $(1, 2)$ is not in the relation.

It is symmetric. But the failure of reflexivity and transitivity means it is not an equivalence relation.

b. The relation S on $\mathbb{R}^2$ defined by the set of ordered pairs of coordinates:

$$S = \left\{(a, b) \;\middle|\; \|a\| = \|b\|\right\},$$

where $\|a\|$ is the distance from a to the origin in $\mathbb{R}^2$ ($\mathbb{R}^2$ is the set of ordered pairs (x, y) where $x, y \in \mathbb{R}$, also known as the set of points in the plane, and the distance from a point (x, y) to the origin is defined as $\sqrt{x^2 + y^2}$.)

[1] You might find this resource here helpful for this question.

In addition to your proof, answer the following: given a fixed point $p \in \mathbb{R}^2$, the collection of all points related to p gives what familiar geometric object? That is, what is $\{x \mid x\ S\ p\}$?

Solution:

We will prove that R is an equivalence relation.

Reflexivity: Consider an arbitrary p in R^n. (p, p) must be in R, as the distance from p to P is the same as, well, the distance from p to P. Thus, R is reflexive.

Symmetry: Consider some (a, b) in R. If (a, b) in R, then the distance from b to P is the same as the distance from a to P. By the definition of R, (b, a) must be in R. Thus, R is symmetric.

Transitivity: Consider some (a, b) and (b, c) in R. If (a, b) in R, then the distance from a to P, b to P, and c to P must all be equal. Thus, the distance from a to P and c to P are equal. Therefore, (a, c) must be in R. Thus, R is transitive.

Because R is reflexive, symmetric, and transitive, it must be an equivalence relation.

R describes a circle (or a point with $p = (0, 0)$) centered about the origin and passing through point p.

c. Let $A = \{a, b, c, d\}$. Let T be the relation on A with the graph:

$$\{(a, a), (b, b), (c, c), (a, b), (b, c), (a, c), (b, a), (c, b), (c, a)\}.$$

Solution:

This is not an equivalence relation because it is not reflexive. The element d is in S, but (d, d) is not in R.

Problem 2

The CS22 plant nursery is now open and quickly growing in popularity! The first customer came in and bought 1 plant. The second bought 8. The third bought 27. The pattern has continued: the kth customer bought k^3 plants.

This is a lot of plants, and to keep our inventory up, we want to predict how many plants in total we will have sold after the nth customer has made their purchase. Call this number of total sales $S(n)$.

Our staff has made some guesses for the value of $S(n)$:

1. $S(n) = n^3 + (n-1)^3$
2. $S(n) = \frac{1}{4}n^2(n+1)^2$
3. $S(n) = 8^{n-1}$

a. Assuming that one of the guesses is correct, which one do you believe? Briefly justify your answer. (You do not need to prove it; that's the next part.)

Solution:

We can rule out 1 and 3 by testing a few values. $S(1)$ should be 1, $S(2)$ should be 9, $S(3)$ should be 36.

The equation in (1) would make $S(3) = 35$, so that one is out. The equation in (3) would make $S(2) = 8$, so that one's out. (2) must be the winner.

b. Prove, using induction, that your chosen formula is correct. You should follow the "recipe" for an induction proof, stating the induction predicate clearly.

Solution:

We prove by induction on n that for every $n \in \mathbb{N}$,

$$\sum_{k=0}^{n} k^3 = \frac{1}{4}n^2(n+1)^2.$$

(The above is our induction predicate P.)

Base case: when $n = 0$, the left hand side of the sum is 0, as is the right hand side. So $P(0)$ holds.

Induction step: suppose $P(n)$ holds, that is, $\sum_{k=0}^{n} k^3 = \frac{1}{4}n^2(n+1)^2$. We will show that $\sum_{k=0}^{n+1} k^3 = \frac{1}{4}(n+1)^2(n+2)^2$ by calculating:

$$\begin{aligned}\sum_{k=0}^{n+1} k^3 &= \left(\sum_{k=0}^{n} k^3\right) + (n+1)^3 \\ &= \left(\frac{1}{4}n^2(n+1)^2\right) + (n+1)^3 \\ &= \frac{1}{4}\left(n^4 + 6n^3 + 13n^2 + 12n + 4\right) \\ &= \frac{1}{4}(n+1)^2(n+2)^2\end{aligned}$$

as desired. This shows $P(n+1)$, so our induction proof is complete.

Problem 3

This problem is a Lean question!

This homework question can be found by navigating to `BrownCs22/Homework/Hw4.lean` in the directory browser on the left of your screen in Gitpod. The comment at the top of that file provides more detailed instructions.

NOTE: the submission method is different this week! Now, you can submit a .pdf file, a .tex file, and a .lean file all to the same assignment. The autograder will check your .lean file as normal, and also make sure you have submitted the .tex source for the rest of your solutions.

Question 3 Autograded on Lean/Gradescope.

Problem 4 (Mind Bender — *Extra Credit*)

Mind Benders are extra credit problems intended to be more challenging than usual homework problems and are an exploration into a topic not covered in lecture. This week, we'll think about induction over sets other than $\mathbb{N}$.

First of all: what is $\mathbb{N}$, exactly? Our definition $\{0, 1, 2, \ldots\}$ wasn't very precise. Here's a more careful attempt. We'll say a set S is *number-like* if it satisfies the following properties:

1. $0 \in S$.
2. For every $n \in S$, $\mathsf{successor}(n) \in S$.
3. The successor function is injective.
4. For every $n \in S$, $\mathsf{successor}(n) \neq 0$.
5. Every element of S is either 0 or the successor of some other element of S. (In other words, rules 1 and 2 are the only ways to "create" natural numbers. In yet more words, the range of successor is $S \setminus \{0\}$.)

We then define $\mathbb{N}$ to be the *smallest* number-like set: that is, the intersection of all number-like sets. Note that 0 is in this intersection, since it must be in every number-like set. Similarly for $\mathsf{successor}(0)$, ...

a. Using this characterization of $\mathbb{N}$, explain why the principle of induction works to prove facts about all natural numbers. (Note: $n + 1$ is another way to write $\mathsf{successor}(n)$.)

Solution:

These properties basically codify the claim that the natural numbers start at 0, go up by successor steps, and don't go anywhere else. This is exactly how proof by induction works: start at 0, follow the "successor chain," and you hit every natural number along the way.

b. We need each of properties 3-5 to make sure that $\mathbb{N}$ has the shape we want.

> **Example**
>
> Suppose we defined $\mathcal{N} = \{0, 1, 2\}$ with
>
> $$\begin{aligned}\mathsf{successor}(0) &= 1\\ \mathsf{successor}(1) &= 2\\ \mathsf{successor}(2) &= 0\end{aligned}$$
>
> $\mathcal{N}$ satisfies properties 1, 2, 3, and 5, but fails 4.

Give an example like this of a set and successor function that satisfies every property *except* property 3, and another example that satisfies every property *except* 5.

Solution:

For every property except 3: let $S = \{0, 1\}$, with $\mathsf{successor}(0) = \mathsf{successor}(1) = 1$.

For every property except 5: let $S = \mathbb{N} \cup \{\infty\}$. ∞ is not the successor of any natural number.

We can give a similar characterization of *lists* of elements of some set T.

1. The empty list [ ] is a list of elements of T.
2. For any $t \in T$ and list L of elements of T, $\mathsf{append}(t, L)$ is a list of elements of T: that is, it's the list we get when we *append* t at the beginning of L.

3-5. Similar to 3-5 above for $\mathbb{N}$.

This may not be a definition of lists that you've seen before. We call it, maybe tellingly, an *inductive* (or *algebraic*) definition.

c. "Translate" properties 3-5 from the $\mathbb{N}$ case so that they make sense in the list case.

Solution:

3: The append function is injective.

4: For every $t \in T$ and list L of elements of T, $\mathsf{append}(t, L) \neq []$.

5: Every list of elements of T is either the empty list, or an element of T appended to a list of elements of T.

d. Lists can be a kind of mathematical object just like numbers: we may want to prove a theorem about all lists of elements of some set T. Analogous to the principle of induction for $\mathbb{N}$, there's a principle of induction for lists. State this principle of induction.

Solution:

Let $P(L)$ be a property of lists of elements of T. To prove $P(L)$ holds for all lists L, we can show two things: $P([])$ holds, and for all t and L, if $P(L)$ holds then $P(\mathsf{append}(t, L))$ holds.

e. We say a *subsequence* is a non-necessarily contiguous subset of a list L. For example, if L were the list $[1, 2, 3, 4, 5]$, then one such subsequence could be $[2, 4, 5]$. Prove, by inducting on the structure of lists, that the number of subsequences of a list of length n (containing all distinct elements) is 2^n.

f. Can you think of any other sets/structures that can be defined "inductively" like this?

Note: these sets show up all the time in certain areas of computer science. Some programming languages *only* allow you to define new datatypes as inductive sets[2] (or "inductive types")! What if you write a complicated program that takes inputs from one of these datatypes, and want to prove that your program has certain behavior? You guessed it: time for induction.

[2] For those who have taken a computer science course that has introduced functional programming, append may look very similar to link or cons!