Ali Awari

aka87

CS 205: Final Exam

Let = {0, 1, 2, . . . , N − 1}, i.e., the integers greater than or equal to 0, and strictly less than N

1. **Define a function by**
2. The domain, range (co-domain), and image is [0, 13):

* More specifically, the domain and range is { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 }
* The image is { 0, 6, 12, 5, 11, 4, 10, 3, 9, 2, 8, 1, 7 }
* Suppose the mapping of by , isn’t injective.
* That would mean there are at least two elements in the domain of such that, when entered into the function would return the same result.
* Since the mapping of the function is a result of modular arithmetic if there is a number in the domain of such that itself is divisible by 13 or such that when multiplied by the constant 6 is a multiple of 13 then the function will not be injective.
* Since 13 is a prime number, it isn’t divisible by any number except itself and 1. An example of a number pair that would prove that isn’t injective would be positive or negative (0, 13). The domain however consists only of positive numbers and goes up to, at most 12.
* Therefore there is a contradiction, is injective.

1. Yes, the function is invertible because it is a one-to-one correspondence

* { 0, 6, 12, 5, 11, 4, 10, 3, 9, 2, 8, 1, 7 } {0, 10, 7, 4, 1, 11, 8, 5, 2, 12, 9, 6, 3}

1. **Define a function by**
2. The domain, and range (co-domain) is [0, 13). The image is { 0, 6, 12, 3, 9 }.

* More specifically, the domain and range is:

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 }

* To reiterate, the image is { 0, 6, 12, 3, 9 }
  + Suppose the mapping of by , isn’t injective.
  + That would mean there are at least two elements in the domain of such that, when entered into the function would return the same result.
  + Since the mapping of the function is a result of modular arithmetic if there is a number in the domain of such that itself is divisible by 15 or such that when multiplied by the constant 6 is a multiple of 15 then the function will not be injective.
  + Since 15 is not prime, since there are numbers that can divide 15; there exists least common multiples of 15 and that constant, if the product of 6 and constant element of the domain are equal to the LCM then is not injective.
  + Assume that g is not surjective, that means that is not mapped to by g.
  + That means that y is not included in the image of g, which implies the cardinality of the image does not equal the cardinality of the domain
  + Since is true, g is not surjective

1. Give a set A ⊂ S15 and B ⊂ S15 such that g is an invertible map from A to B. Find the largest A, B you can.
   * A = { 0, 1, 2, 3, 4 }, B = { 0, 6, 12, 3, 9 }

Given a function f : S → S (unrelated to the above), a value x ∈ S is a fixed point or has order 1 if x = f(x). Similarly, x has order 2 if x = f(f(x)), and order 3 if x = f(f(f(x))). A point x has order k if applying f k-times to x yields x. If x never returns to x under repeated applications of f, then x has infinite order, or is transitory.

1. **For any N > 1, give an example of an f: that has no fixed points.**

f(0) = 0 : 1 -> 2 -> 3…N

f(1) = 1 : 2 -> 3 -> 4…N

f(2) = 2 : 3 -> 4 -> 5…N

1. **For any N > 1, give an example of an f: that has only a single point of finite order.**

f(0) = 0 : 0 -> 0 -> 0…0

f(1) = 0 : 1 -> 0 -> 0…0

f(2) = 0 : 2 -> 0 -> 0…0

1. **Argue that for any N > 1, given f: , if x has finite order, the x has an inverse under f, i.e., if every point has a finite order, then f is invertible. Consider small examples first.**

f(0) : 0

f(1) : 3

f(2) : 1

f(3) : 2

* If every x has a finite order under f, then x is a fixed point, since the number of elements are the same, this implies that x must be fixed to one point on the codomain, implying that x is surjective and injective (bijective). All bijections are invertible.

Suppose you had a program that, given an image file as input, returned a tag from a finite set of tags Tags = {cat, dog, orange, . . . , UFO, CannotBeDetermined}, describing the contents of that image (a standard computer vision problem).

1. **Describe this program as a function, mapping one set to another set. What is the domain, what is the range? Be as precise as you can be. Is this function invertible?**

* Consider A the set of input image files, B the finite set of tags, being the map between :
* The domain of this function is all the possible input images (including images which contain the same object in a different context)
* The range is the finite set of tags given to us by the program.
* This function is not invertible since images with the same object but different context are mapped to the same tag, as well as images which are not able to be identified are all set to the default element in the range “CannotBeDetermined” implying that the function is not injective, therefore not invertible.

1. **Suppose that the program were restructured so that it returned a collection of tags describing the contents of the image. For instance, a picture of a dog and a cat sitting together might return {dog, cat}. Describe this program as a function, mapping one set to another set. What is the domain, what is the range? Be as precise as you can be. Is this function invertible?**

* Consider A the set of input image files, B the finite set of tag sets, being the map between :
* For all sets in B, represented as y, there exists an element in A such that x maps to y.
* The domain of this function is all the possible input images (including images which contain the same object, or multiple objects, in a different context)
* The range is the finite set of tag sets in which the program converts objects in an image into a tag set, this converted object may or may not be able to be identified.
* This function is not invertible for the same reasons as number 6.

**Bonus**:

This argument is independent from computer speed, architecture / design because it is a fundamental limitation of computation. Since programs are finite there is a limitation to the amount of different numbers they can compute using arithmetic while there is an infinite amount of numbers, therefore there exists non-computable numbers.

**1) Formally prove that if and then**

* , when
* , when

**2) True or false - give a mathematical justification:**

a)

* **True**, functions may be bounded by a higher order, though it wouldn’t be too helpful in real world applications.
* **True**
* Bound all values by:
* **False**
* Bound all values:
* **True**
* Multiply both sides by
* **False**
* **True**, assuming that an algorithm must have at least a single step, it is lower bounded by .

g)

* **True**

h)

* **True**
* Prove that:
* For every

i)

* **False**
* Prove that:
* For every

j)

* **True**

Recall the idea of merge sort: to sort a list, divide a list in two, sort the two halves, and merge them to form a sorted whole. In class, we gave an argument that the complexity of merge sort on a list of N elements could therefore be described as M(N) = M(N/2) sort the left half + M(N/2) sort the right half + N merge the two halves = 2M(N/2) + N. (1) Noting that M(1) = 1, since sorting a list of size 1 is easy, this led to an overall complexity of M(N) = O(N ln N). Your good buddy suggests the following: If merge sort gets such good performance dividing the list into two halves and merging them, imagine how fast a merge sort would be that split the list to sort into three parts, sorted them, then merged the result.

3) Like the recursive relation above, give a rough description of the overall worst case complexity of this tri-merge sort

TERNARY SORT V.S. BINARY SORT

4) In terms of big-O, which approach has the smaller complexity?

5) In your opinion, is it worth the additional effort and overhead it would take to implement this approach? Justify.

<https://www.geeksforgeeks.org/binary-search-preferred-ternary-search/>

<https://www.geeksforgeeks.org/3-way-merge-sort/>

Question 4: This is a straight extension of the material I spent basically a whole lecture on and posted a whole batch of notes on. If you can solve 3x + 5y = D for any value of D, you should be able to solve 3x + 5y = 1 - 7z for any given value of z.

Question 4)  
  
In class/notes/preamble for the problem, we considered solving equations like 3x + 5y = D for various values of D. We connected this to the problem of solving 3x + 5y = 1 and generalizing to other D values.  
  
In this problem, you're asked to generalize this to 3x + 5y + 7z = 1. While this may seem initially daunting, if I told you to start by assuming that z = 0, you would have no trouble simplifying this to 3x + 5y = 1, and applying previous results. If I told you that z = -5, you'd have no/little trouble simplifying this to 3x + 5y = 36, and again applying previous results. If you can solve it for any specific value of z, how could you generalize this approach to think about arbitrary values of z?  
  
For the system of equations - once you have gleaned all you can about how x, y, and z relate to each other from Equation 1, how can you use Equation 2 to further relate the variables and their forms?  
  
A separate but related point: How can you argue that you have generated / parameterized /all/ solutions to the equation? What this boils down to, as a small example, is  
  
if I have the congruence, 7x === D (mod 11), and I solve for x given a value for D, how can I be sure that my solution is the only solution? One way to frame this is to imagine defining the function f(x) = 7x (mod 11). Operating over the numbers 0, 1, 2, ..., 10, what you're really asking is, for a given value of D (mod 11), solve the equation  
  
f(x) = D  
  
which gives  
  
x = f^(-1)(D) [f inverse of D]  
  
When can you be sure that this inverse exists or is unique? What does that say about f?  
  
That's all for today. Please continue to ask questions as they arise.

- The notes walk through a pretty thorough introduction to these ideas, I think - essentially introducing an approach to solving congruences like 3x === D  (mod 5) based on the solution to 3x === 1 (mod 5). How could the approach taken there be generalized?