MAT 305: Review #6

June 18, 2014

Directions: The usual counsels apply.

Part I
Modifying an already-written function

1. In class, we wrote a function, `method_of_bisection`, that approximates the root of a function to 2 decimal places. Rewrite the function so that it approximates the root of a function to d decimal places, where d is an argument specified by the user.

Part II
Functions with definite loops

On a previous assignment, you had to decide whether a given set (the *quaternions*) was a group. This required a lot of calculations that you had to request by hand. This time, you'll use functions and loops, so that you can check this for arbitrary sets and operations.

2. A finite set S is "a group under an operation $\otimes$" if it satisfies the four properties of:

   - *closure,* that is, $x \otimes y \in S$ for all $x, y \in S$;
   - *associative,* that is, $x \otimes (y \otimes z) = (x \otimes y) \otimes z$ for all $x, y, z \in S$;
   - *identity,* that is, we can find $\iota \in S$ such that $x\iota = x$ and $\iota x = x$ for all $x \in S$;
   - *inverse,* that is, for any $x \in S$, we can find $y \in S$ such that $xy = yx = \iota$.

   In Figures 1, 2, 3, and 4 you will find pseudocode for a computer program to test each property. Implement three of your algorithms as a Sage function. The operation should be used as a function with two inputs x and y; you can use it by invoking `set_operation(x,y)`. **I will do an example of one property in class, which should make the others easier.**

3. Test your answers on each of the following possibilities:

   (a) S is the quaternions, and $\otimes$ is matrix multiplication. (This should satisfy all properties.)

   (b) S is the set of integers from `0` to 11, and $\otimes$ is addition, *modulo 12*. (This should satisfiy all properties.)

   (c) S is the set of integers from `0` to 11, and $\otimes$ is multiplication, *modulo 12*. (This should satisfy all properties, *except* the inverse property.)

4. Write *one* function that, when given a finite set S and an operation $\otimes$, determines whether S is a group under $\otimes$. *You may (and should) invoke the programs you've already written.*

5. Use your program(s) to answer the following.

   (a) Use your programs to determine which of the following satisfy all four properties.

      (i) $\mathbb{N}_5 = \{1,2,3,4\}$, where the operation is *multiplication*, modulo 5.

      (ii) Choose 5 values of n, larger than 11 but smaller than 20. For each value of n, repeat (i) for $\mathbb{N}_n = \{1,2,\ldots,n-1\}$. Note that the operation changes to multiplication *modulo* n, rather than modulo 5.

      (iii) $\mathcal{P} = \{1,x,x+1\}$, where the operation is multiplication modulo 2 *and* modulo x^2+1. To set this up,

      - define a "base ring," $R = \mathbb{Z}_2$ (`sage: R = GF(2)`)
      - define a "quotient ring," $S = \mathbb{Z}_2/\langle x^2+1\rangle$ (`sage: S = (R[x]).quo(x^2+1)`)
      - define the set $\mathcal{P}$ (`sage: P = [S(1),S(x),S(x+1)]`)
      - define the operation $\otimes$ (`sage: def op_mul(a,b): return a*b`)

      When you test identity, use `S(`ι`)` to make sure your guess for ι has the right form.

      (iv) $\mathcal{Q} = \{1,x,x+1\}$, where the operation is multiplication modulo 2 *and* modulo x^2+x+1. To set this up, repeat the above, but replace `x^2+1` by `x^2+x+1`.

      (v) $\mathcal{F} = \{x,x^2,\sqrt{x}\}$, where the operation is composition of functions. To set this up,

      - define the set $\mathcal{F}$ (`sage: F = [x, x^2, sqrt(x)]`)
      - define the operation $\otimes$ (`sage: def op_compx(a,b): return a(x=b)`)

   To check yourself: (i) satisfies all properties; (ii) sometimes fails inverse, but otherwise satisfies all; (iii) fails closure and inverse; (iv) satisfies all; (v) satisfies *only* identity.

   (b) Look at the results of (i) and (ii). What characteristic was shared by all the sets that satisfied all four properties?

   (c) Try multiplying some of the elements of (iii). What products surprise you *most?* All of them should be somewhat surprising, but two are more surprising than others.

```
algorithm is_closed
  inputs
    S, a finite set
    ⊗, an operation on S
  outputs
    whether S is closed under ⊗
  do
    Let result = true
    for s ∈ S
      for t ∈ S
          if s t ∉ S
            Let result = false
    return result
```

Figure 1: Pseudocode for Closure

```
algorithm is_associative
  inputs
    S, a finite set
    ⊗, an operation on S
  outputs
    whether S is associative under ⊗
  do
    Let result = true
    for s ∈ S
      for t ∈ S
        for u ∈ S
          if (s t) u ≠ s (t u)
            Let result = false
    return result
```

Figure 2: Pseudocode for Associative

```
algorithm has_identity
  inputs
    S, a finite set
    ⊗, an operation on S
  outputs
    true and the identity of S under ⊗, if it has one
    false otherwise
  do
    Let identity = false
    for ι ∈ S
      Let maybe_identity = true
      for s ∈ S
        if ιs ≠ s or sι ≠ s
          Let maybe_identity = false
      if maybe_identity
        return true, ι
    return false
```

Figure 3: Pseudocode for Identity

```
algorithm satisfies_inverse
  inputs
    S, a finite set
    ⊗, an operation on S
    ι, the identity of S under ⊗
  outputs
    true iff S satisfies the inverse property under ⊗
  do
    for s ∈ S
      Let has_inverse = false
      for t ∈ S
        if st = ι
          Let has_inverse = true
      if not has_inverse
        return false
    return true
```

Figure 4: Pseudocode for Inverse