Sage Exercises
Math 434, Spring 2010
Dr. Beezer

Submit these exercises in a Sage worksheet attached to an email sent to the course's `privacyport` address. Please include your name near the beginning of the worksheet and use a descriptive filename for the attachment (names, chapter combination?). These are due prior to class on the day we have a problem session for the chapter.

# Chapter 14, Rings

These exercises are intended to give you some experience with rings and fields in Sage, in preparation for subsequent chapters.

1. Define the two rings $\mathbb{Z}_7$ and $\mathbb{Z}_{12}$ with the commands `R=Integers(7)` and `S=Integers(12)`. For each ring, locate and use the relevant command to determine: if the ring is finite, if it is commutative, if it is an integral domain and if it is a field. Then use single Sage commands to find the order of the ring, list the elements, and output the multiplicative identity (i.e. 1, if it exists).

2. Define `R` to be the ring of integers, $\mathbb{Z}$, by executing `R=ZZ` or `R=Integers()`. A command like `R.ideal(4)` will create the principal ideal $\langle 4 \rangle$. The same command can accept more than one generator, so for example, `R.ideal(3,5)` will create the ideal $\{a \cdot 3 + b \cdot 5 \mid a, b \in \mathbb{Z}\}$. Create several ideals of $\mathbb{Z}$ with two generators and ask Sage to print each as you create it. Explain what you observe.

3. Create a finite field $F$ of order 81 as follows: `F.<t>=FiniteField(3^4)`.

   (a) List the elements of $F$.

   (b) Get the generators of $F$ with `F.gens()`.

   (c) Get the first generator of $F$ and save it as `u` with `u=F.0` (alternatively, `u=F.gen(0)`).

   (d) Compute the first 80 powers of `u` and comment.

   (e) The generator you have worked with above is a root of a polynomial over $\mathbb{Z}_3$. Obtain this polynomial with `F.modulus()` and use this observation to explain the entry in your list of powers that is the fourth power of the generator.

4. A ring of polynomials, $P$, in the variable $z$, with coefficients from the ring $R$ can be built in Sage with the command `P.<z>=R[]`. (Or alternatively, `P=R['z']`, or `P=PolynomialRing(R,'z')`.

   Build and analyze a quotient ring as follows:

   (a) Build a ring $P$ of polynomials in $z$ over $Z_7$: `P.<z>=(Integers(7))[]`

(b) Build a principal ideal $K$ with the polynomial $z^2 + z + 3$: `K=P.ideal(z^2+z+3)`

(c) Form the quotient ring of $P$ by $K$, $H$: `H=P.quotient(K)`

(d) Use Sage to verify that $H$ is a field.

(e) As in the previous exercise, get a generator and examine the proper collection of powers of that generator.

# Chapter 15, Polynomials

1. In this exercise you will work with the polynomial $x^3 - 3x + 4$. Consider this polynomial as an element of a polynomial ring over (a) the finite field $\mathbb{Z}_5$, (b) a finite field with 125 elements, (c) the rationals, (d) the real numbers and (e) the complex numbers. In each case use Sage to factor the polynomial as fully as possible. It should be straightforward to create and/or use each of these fields, and then make sure you create the polynomial in the right ring. For a Sage polynomial `poly` the commands `poly.factor()` and `poly.roots()` could be useful.

2. "Conway polynomials" are a collection of irreducible polynomials over $\mathbb{Z}_p$ that Sage (and other software) uses to build maximal ideals in polynomial rings, and thus quotient rings that are fields. Roughly speaking, they are "canonical" choices for each degree and prime. The command `conway_polynomial(p,n)` will return a database entry that is a polynomial of degree $n$ over $\mathbb{Z}_p$.

   Execute the command `conway_polynomial(5,4)` to obtain an allegedly irreducible polynomial of degree 4 over $\mathbb{Z}_5$: $x^4+4x^2+4x+2$. Use a combination of sensible deductions and brute-force to allow Sage to help you conclude this polynomial is really irreducible (or perhaps locate a bug in Sage). Two double-loops and one single-loop should be sufficient. Loops nested six-deep is totally brute-force, with no deductions applied to the problem (so be more clever than this). Don't forget to use coercion liberally if necessary.
   `http://www.math.rwth-aachen.de/~Frank.Luebeck/data/ConwayPol/index.html`

3. Define $p = p(x) = -6x^5 - x^4 - 7x^2 - 13x - 2$ and $q = q(x) = 5x^5 - 7x^4 + 4x^3 + 2x^2 - 2x + 1$ as polynomials over the integers $\mathbb{Z}$. In Sage compute their GCD with `gcd(p, q)`, then find the extended GCD with `xgcd(p, q)`. Do you believe these results? Is the situation different if you view these polynomials over the rationals $\mathbb{Q}$?.

# Chapter 16, Integral Domains

Nothing assigned.

# Chapter 17, Lattices and Boolean Algebras

1. Build a random poset using the command `R = Posets.RandomPoset(20,0.05)`. Use `R.plot()` to get an idea of what you have built.

   (a) Illustrate the use of the `is_lequal(), is_gequal()` commands to determine if two elements are related or incomparable.

   (b) Find the smallest and largest elements of your poset using the `minimal_elements()` and `maximal_elements()` commands.

   (c) See if your poset is a lattice by attempting to construct a lattice from `R` with the command `LatticePosetR`.

   (d) Find a linear extension of your poset and use it to provide a concise explanation of what a linear extension is. One place to start on this is: `http://planetmath.org/encyclopedia/ExtensionOfAPoset.html`.

2. Construct the example from class that is the poset on the positive divisors of 36 with divisiblity as the relation. Read about creating a poset from documentation for the `Poset()` command. Python's `lambda` command can be the quickest and easiest way to do this. Otherwise, you might just enter all the pairs of elements which form the edges in the Hasse diagram. As above, construct the lattice from this poset by feeding the poset into the `LatticePoset()` command. Use Sage commands to determine

   (a) the one and zero element (`top()`, `bottom()`),

   (b) all the elements of the lattice that are complements of each other (you may need the `list()` command to interpret the output),

   (c) if the lattice is distributive.

3. Build the prototypical Boolean algebra using `Posets.BooleanLattice(4)` (i.e. subsets of a 5-set). Build the poset whose elements are partions of the integer 5, where the relation is one partition being a refinement of the other, using `Posets.IntegerCompositions(5)`. Plot each to see that they are very similar. Use the method `hasse_diagram()` on each poset to get back a directed graph. Then use the `is_isomorphic()` method to see that the two Hasse diagrams really are the same. Can you construct an explicit isomorphism between the two Boolean algebras?

4. Construct several diamond lattices with `Posets.DiamondPoset(n)` by varying the value of `n`. Give answers, with justifications, to these questions **for general** $n$ based on observations obtained from experiments with Sage.

   (a) Which elements have complements and which do not, and why?

   (b) How many antichains are there?

   (c) Is the lattice distributive?

# Chapter 18, Vector Spaces

1. Build a finite field of a desired size in the usual way, using `F = GF(p^n, 'a')`. Then construct the (multiplicative) group of all invertible (nonsingular) $m \times m$ matrices over this field with the command `G = GL(m, F)` ("the general linear group"). What is the order of this group in general? Your answer should be a function of $m$, $p$ and $n$ and should include an explanation of how you come by your formula (i.e. something resembling a proof).

   Hints: `G.order()` will help you test and verify your hypotheses. Small examples in Sage (listing all the elements of the group) might aid your intuition—which is why this is a Sage exercise. Small means $2 \times 2$ and $3 \times 3$ matrices and finite fields with $2, 3, 4, 5$ elements, at most. Results don't really depend on each of $p$ and $n$, but rather just on $p^n$.

   Realize this group is interesting because it contains representations of all the invertible (i.e. 1-1 and onto) linear transformations from the (finite) vector space $F^m$ to itself.

2. Given two subspaces $U$ and $W$ of the vector space $V$, their sum $U + W$ can be defined as

$$U + W = \{u + w \mid u \in U,\ w \in W\}$$

   Notice, this is not the direct sum of your text, nor the `direct_sum()` method in Sage. However, you can build this in Sage as follows. Grab the bases of $U$ and $W$ individually, as lists of vectors. Join the two lists using Python syntax—just use a plus sign in-between. Now build the new vector space that is the sum by creating a subspace of $V$ with the `.subspace()` method.

   Build a largish vector space over the rationals (`QQ`), where "largish" means perhaps dimension 7 or 8 or so. Construct a few subspaces and compare their individual dimensions with the dimensions of the intersection of $U$ and $W$ ($U \cap W$, `.intersection()` in Sage) and the sum $U + V$. Form a conjecture relating these dimensions based on your (nontrivial) experiments.

3. The matrix $C$ below is nilpotent. Find the index. Find a matrix $S$ so that $S^{-1}CS$ is block diagonal with Jordan blocks as the blocks. Use Sage only to compute powers of matrices, null spaces (the `.right_kernel()` command) and simple computations with vectors. In particular, do not use any eigenvector commands.

$$C = \begin{bmatrix} 5 & 4 & 9 & 15 \\ 2 & 1 & 3 & 6 \\ -2 & -1 & -3 & -6 \\ -1 & -1 & -2 & -3 \end{bmatrix}$$

4. Find a basis for $\mathbb{C}^6$ so that a matrix representation of $T$ will be in Jordan canonical form. It should be possible to avoid fractions in your work. Use Sage only to compute powers of matrices, null spaces (the `.right_kernel()` command), eigenvalues and simple computations with vectors. In particular, do not use any eigenvector commands, and of course, do not use

Sage to simply get the Jordan canonical form itself.

$$T : \mathbb{C}^6 \to \mathbb{C}^6, \quad T(\mathbf{x}) = A\mathbf{x}$$

$$A = \begin{bmatrix} -3 & -1 & -4 & -1 & -3 & -3 \\ 5 & 6 & 5 & 4 & 3 & 4 \\ 0 & -3 & 1 & 0 & 3 & 2 \\ -8 & -13 & -6 & -9 & -4 & -6 \\ 7 & 11 & 7 & 5 & 2 & 5 \\ 1 & 2 & -1 & 3 & 1 & 0 \end{bmatrix}$$

# Chapter 19, Fields

You will want to use coercion liberally in this set of exercises. For example, if `K` is a field that you have defined in Sage, then will create `L` as the set (ring) of all polynomials in the variable `x` with coefficents in `K`. Now suppose that `K` is a field (like the rationals) that contains the integers. Then a command like `poly = L(x^2-x+1)` will insure that the coefficients of the polynomial are viewed as elements of $K$. This is a fallback when you absolutely need to be sure your polynomials are considered over the proper field. The syntax `L.<x>=K[]` can often make this unnecessary.

1. Create a finite field $F$, of order $3^5 = 243$ using Sage's `FiniteField[]` command, specifying the irreducible polynomial (over $\mathbb{Z}_3$) to be $p(x) = x^5 + 2 * x^4 + 1$ using the `modulus` keyword. this is where you want to be certain that $p(x)$ is of the right type — in other words, it needs to be considered as a polynomial over $\mathbb{Z}_3$.

   Now consider $p(x)$ as a polynomial over $F$ (i.e. inform Sage of this, which may require also changing the name of the variable). Then factor $p(x)$ over $F$ and report the distinct roots.

2. This problem builds on the previous one. Create the ring of polynomials over $\mathbb{Z}_3$ in the variable $x$ and create the principal ideal $\langle x^5 + 2x^4 + 1\rangle$. Then form the quotient ring by this ideal. This will be a field (the ideal is maximal since the polynomial is irreducible), and since the polynomial employed is the same, will be isomorphic to the field of the previous exercise by a nearly-trivial isomorphism.

   According to this isomorphism, convert the roots of $p(x)$ found earlier into elements of this quotient ring. Note the cosets are denoted with the "bar notation." Test each root of $p(x)$ with the relevant computation in the quotient ring.

3. In the section in this chapter relying on techniques from linear algebra, Judson proves that every finite extension is an algebraic extension. This exercise will help you understand this proof.

   Extensions of the rationals are known as "number fields." The polynomial $r(x) = x^4+2x+2$ is irreducible over the rationals (Eisenstein's criterion with prime $p = 2$). Use the `NumberField` command to create an extension of the rationals where $r(x)$ has a root. According to the theorem *every* element of this extension is algebraic, i.e. is a root of *some* polynomial over the rationals. Sage's `minpoly()` method will compute such a polynomial (the minimal polynomial) for any element of the extension.

   For this exercise, suppose the root of $r(x)$ in the extension is $a$, and then find the minimum polynomial of $t = 3a + 1$ with the following method. According to the proof, the first five powers of $t$ (start counting from zero) will be linearly dependent. Why? So a nontrivial relation of linear dependence on these powers will provide the coefficients of a polynomial with $t$ as a root. Compute these powers, form the right linear system, and suitably interpret its solutions. Hint: given an element of the number field, which will necessarily be a polynomial in the generator $a$, the `vector()` method on the element will provide the coefficients of this polynomial in a list.

4. Construct the splitting field of $q(x) = x^4 + x^2 + 1$.

5. Form the number field, $K$, which contains a root of the polynomial $q(x) = x^3 + 3x^2 + 3x - 2$. Verify that $q(x)$ factors, but does not split, over $K$. With $K$ now as the base field, form an extension of $K$ where the quadratic factor of $q(x)$ has a root. With your polynomial formed properly over the correct field, this can be accomplished simply with the `K.extension()` command (using the right arguments, etc). Call this second extension in the tower $L$.

   Flatten your tower by using the command `M.<b>=L.absolute_field()`, which will create $M$ as a straight-out extension of $\mathbb{Q}$ (whereas $L$ is considered a "relative" extension of $K$. Type just `M` to see the defining polynomial for $M$ (or use the appropriate command). From this, you should be able to use high-school algebra to write the generator $b$ in a fairly simple form.

   $M$ should be the splitting field of $q(x)$. To see this, start over, and build a number field, $P$, from scratch using the simple expression you just found. You should be able to do this with a command like `P.<c>=QQ[<simple-expression>]` and Sage will do the rest for you. Then factor $q(x)$ as a polynomial from $\mathbb{Q}[x]$, but within the extension $P$, to see that the polynomial really does split. Using this factorization, and your simple expression for $b$ write simplified expressions for the three roots of $q(x)$.

# Chapter 20, Finite Fields

1. Create the finite field $GF(5^2)$ and then factor $p(x) = x^{25} - x$ over this field. Comment on what is interesting about this result and why it is not a surprise.

2. This problem investigates the "seperableness" of $\mathbb{Q}(\sqrt{3}, \sqrt{7})$. Use the `NumberFieldTower` command to quickly build a tower of degree 2 extensions, adjoining one root at a time. Beginning with something like `N.<a,b>` will allow you to have easy-to-recognize names for each root (make sure you know which is which). Now, "flatten" your tower with the `absolute_field()` method. For the following, suppose this flattened version is named `L` with a generator you've named `c`. The command `fromL,toL = L.structure()` will give you back two bijections between `N` and `L` that will allow you to translate between expressions in terms of `a, b` and in terms of `c`.

   (a) Translate the generator of `L` to an expression in the generators of `N`. Use this to write $\mathbb{Q}(\sqrt{3}, \sqrt{7})$ as a simple extension (in math-speak, not Sage-speak).

   (b) Construct a non-trivial "random" element of `L` using all the basis elmeents of $L/\mathbb{Q}$, and ask Sage to create the minimal polynomial of the element. Take the resulting polynomial, which you know has the random element as a root, and factor it — but do the factorization in `N`. Write the roots of this polynomial in math-speak.

   (c) Repeat the prior exercise with another "random" element. If the extension $\mathbb{Q}(\sqrt{3}, \sqrt{7})$ is seperable, then any such factorization should always produce distinct linear factors.

   (d) Save your worksheet. Now see if you can design an element with a minimal polynomial whose factorization stresses (crashes?) Sage.

3. Exercise 20 in this chapter describes the "Frobenius Map," an automorphism of a finite field. If `F` is a finite field in Sage, then `End(F)` will create a list of "endomorphisms" (i.e. homomorphisms from a set to itself). You can extract these mappings from the list and then use them as functions with inputs from the relevant finite field.

   (a) Work Exercise 20 to gain an understanding of how and why the Frobenius mapping is a field automorphism. (Don't include any of that in your worksheet, but understand that the following will be much easier if you do this problem first.)

   (b) For some small, but not trivial, finite fields locate the Frobenius map in Sage's output. Small might mean $p = 2, 3, 5, 7$ and $3 \leq n \leq 10$, with $n$ prime vs. composite. You'll want to use Sage as a fancy calculator to help you with this.

   (c) Once you have recognized the Frobenius group, see if you can conjecture a precise description of every automorphism of a finite field. How might you now establish that these **endomorphisms** really are **automorphisms**?

   (d) Characterize the group of all field automorphisms, and describe the special status of the Frobenius map. This could be the basis of a better justification of the endomorphisms being automorphisms.

(e) Given a field automorphism, $\tau$, for a field extension $E/F$, a central topic of the next chapter will be the fixed field of $\tau$,

$$K = \{b \in E \mid \tau(b) = b\}$$

For each automorphism of $E = GF(3^6)$ identify the fixed field of the automorphism. Since we know a lot about the structure of subfields of a finite field, it is enough to just determine the order of the fixed field to be able to identify the subfield.

# Chapter 21, Galois Theory

1. Find the degree of the splitting field (and hence the order of the Galois group) for the polynomial $x^7 - 7x + 3$. The first two or three steps of this computation will go fairly smoothly. The last two or three steps will each take about an hour, so you should probably run on `sage.ups.edu`, and not at the last minute when everybody else is doing the same.

   This polynomial is derived from an "Elkies trinomial curve," a hyperelliptic curve (below) that produces polynomials with interesting Galois groups:

   $$y^2 = x(81x^5 + 396x^4 + 738x^3 + 660x^2 + 269x + 48)$$

   In this problem the resulting Galois group is $PSL(2, 7)$, a simple group. If $SL(2, 7)$ is all $2 \times 2$ matrices over $\mathbb{Z}_7$ with determinant 1, then $PSL(2, 7)$ is the quotient by the subgroup $\{I_2, -I_2\}$. It is the second-smallest non-abelian simple group (after $A_5$).

2. Build the splitting field of $p(x) = x^3 - 6x^2 + 12x - 10$ and then determine the Galois group of $p(x)$ as a concrete group of explicit permutations. Build the lattice of subgroups of the Galois group, again using the same explicit permutations. Using the fundamental theorem of Galois theory, construct the subfields of the splitting field. Include your supporting documentation in your submitted Sage worksheet. But also, submit in class a written paper component of this assignment containing a complete layout of the subgroups and subfields, written entirely in math-speak and with no Sage-speak, designed to illustrate the correspondence between the two. All you need here is the graphical layout, suitably labeled — the Sage worksheet will back up your work.

3. The polynomial $x^5 - x - 1$ has all of the symmetric group $S_5$ as its Galois group. Because $S_5$ is not solvable, we know this polynomial to be an example of a quintic polynomial that is not solvable by radicals. Unfortunately, asking Sage to compute this Galois group (or evidence for it) takes about 24 hours of computation if we ask Sage to factor polynomials over various intermediate fields and about 6 hours if we are bit more clever. So this exercise will simulate that experience with a slightly smaller example.

   Consider the polynomial $p(x) = x^4 + x + 1$. Build the splitting field of $p(x)$ one root at a time. Create an extension, factor there, discard linear factors, use the remaining irreducible factor to extend once more. Rinse and repeat. Do the final extension with the "leftover" linear factor, even though this is kind of silly and Sage will not display your final adjoined root easily, though Sage will let you reference this final root.

   (a) Factor the original polynomial over the final extension field in the tower. What is boring about this factorization in comparison to some other examples we have done?

   (b) Construct the full tower as an absolute field over $\mathbb{Q}$. From the degree of this extension and the degree of the original polynomial, infer the Galois group of the polynomial.

   (c) Using the bijections that allow you to translate between the tower and the absolute field (from the `.structure()` method), choose one of the roots (any one) and express it in terms of the single generator of the absolute field. Then reverse the procedure and express the single generator of the absolute field in terms of the roots in the tower.

(d) Compute the group of automorphisms of the absolute field (but don't display the whole group in what you submit). Take all four roots (including your silly one from the last step of the tower construction) and apply each field automorphism to the four roots (creating the guaranteed permutations of the roots). Comment on what you see.

(e) There is one nontrivial automorphism that has an especially simple form (it is the second one for me) when applied to the generator of the absolute field. What does this automorphism do to the roots of $p(x)$?

(f) Consider the extension of $\mathbb{Q}$ formed by adjoining just one of the roots. This is a subfield of the splitting field of the polynomial, so is the fixed field of a subgroup of the Galois group. Give a simple description of the corresponding subgroup using language we typically only apply to permutation groups.

(g) Go back to the quintic discussed in the introduction to this problem. Create the first two intermediate fields by adjoining two roots (one at a time). But instead of factoring at each step to get a new irreducible polynomial, *divide* by the linear factor you *know* is a factor. In general, the quotient might factor further, but in this exercise presume it does not. In other words, act as is if your quotient by the linear factor is irreducible. If it is not, then the `NumberField` command should complain (which it won't).

After adjoining two roots, create the extension producing a third root, and do the division. Finally dividing the resulting quadratic will take time (six hours on my desktop). You can try it, or at this stage you can argue (assuming the quadratic is irreducible) that you have enough evidence to establish the order of the Galois group, and hence can determine *exactly* which group it is.