Sage Exercises
Math 433, Fall 2009
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 0

This exercise is just about making sure you know how to use Sage. Login to a notebook server (UPS or UW) and create a new worksheet. Do some non-trivial computation, maybe a pretty plot or some gruesome numerical computation to an insane precision. Maybe include some nicely formatted text or LaTeX using the embedded word-processor (shift-click when a blue bar appears between cells). Create a copy of the worksheet on your own machine by using the "File" selector and then choose "Download to a file." (These are options on the Sage worksheet, not the similar commands for your web browser.) Be sure to use a filename with no spaces and an `.sws` extension. Also, please include your name near the beginning of the worksheet. Then submit the worksheet as an attachment to an email.

Chapter 1

This exercise is about investigating basic properties of the integers, something we will frequently do when investigating groups. Use the Sage group theory primer (link on the course page) to help with some of this. Use the editing capabilities in a Sage worksheet to annotate and explain your work.

1. Use the `next_prime()` method to construct two different 8-digit prime numbers.
2. Use the `is_prime()` method to verify that your primes are prime.
3. Verify that the greatest common divisor of your two primes is 1.
4. Find two integers that make a "linear combination" of your primes equal to 1. Include a verification of your result.
5. Determine a factorization into powers of primes for $b = 4598037234$.
6. Write statements that show that b is (i) divisible by 7, (ii) not divisible by 11. Your statements should simple return `True` or `False` and be sufficiently general that a different value of b or different candidate prime divisors could be easily used.

Chapter 2

These exercises are about getting comfortable working with groups in Sage.

1. Create two groups with the commands `CyclicPermutationGroup(6)` and `SymmetricGroup(3)` and give them names of your choosing. We will understand these constructions better shortly, but for now just understand that they are both groups.

2. Check that the groups have the same size with the `.order()` method. Determine which is abelian, and which is not, by using the `.is_abelian()` method.

3. Use the `.cayley_table()` method to create the Cayley table for each group.

4. Write a nicely formatted discussion (Shift-click on a blue bar to bring up the mini-word-processor, use dollar signs to embed bits of TEX) identifying differences between the two groups that are discernible in properties of their Cayley tables. In other words, what is *different* about these two groups that you can "see" in the Cayley tables?

5. For each group, list the elements of one non-trivial subgroup (not just the identity, and not the whole group). Do *not* use the `x0, x1, x2...` notation. Read the documentation for the Cayley table command, employ the `.list()` method and describe your subgroup as actual elements of the group.

6. Optional: See if you can actually construct your subgroup within Sage so that the `.is_subgroup()` command returns `True`. (Hint: extract the right elements and use all of them as generators of a subgroup.)

Chapter 3

These exercises are about the group of units mod n, $U(n)$, which is sometimes cyclic, sometimes not. There are some commands in Sage that will answer some of these questions very quickly, but instead of using those now, just use the basic techniques described.

1. Execute the command `U=Integers(40)` to create the set $\{0, 1, 2, \ldots, 39\}$. This is a group under addition mod 40, which we will ignore. Instead we are interested in the subset of elements which have an inverse under *multiplication* mod 40. Determine how big this subgroup is by executing the command `U.unit_group_order()`, and then obtain a list of these elements with `U.list_of_elements_of_multiplicative_group()`.

2. You can create elements of this group with syntax like `a = U(7)`. (Don't confuse this with our mathematical notation $U(40)$.) This will tell Sage that you want to view 7 as an element of U, subject to the corresponding operations. Determine the elments of the cyclic subgroup of U generated by 7 as follows:

   ```
   for j in range(16):
       print a^j
   ```

   What is the order of 7 in $U(40)$?

3. The group $U(49)$ is cyclic. Using only the tools described above, use Sage to find a generator for this group. Now using *only* theorems about the structure of cyclic groups, describe each of the subgroups of $U(49)$ by specifying its order and by giving a generator. Do not repeat any of the subgroups, in other words, present each subgroup *exactly* once. You can use Sage to check your work on the subgroups, but your answer about the subgroups should rely only on theorems and be a nicely written paragraph with a table, etc.

4. The group $U(35)$ is not cyclic. Again, using only the tools above, use computations to provide irrefutable evidence of this. How many of the 16 subgroups of $U(35)$ can you list?

5. Again, using only the simple tools outlined, explore the structure of $U(n)$ and see if you can formulate an interesting conjecture about some basic propery of this group. (Yes, this is a *very* open-ended question, but this is ultimately the real power of Sage.)

Chapter 4

These exercises are about becoming familiar with permutation groups in Sage. There is more information in the Sage Group Theory Primer. Permutation groups are perhaps the most concrete way to work with finite groups in Sage, so we will see them throughout the rest of the course.

1. Create the full symmetric group S_{10} with the command `G=SymmetricGroup(10)`.

2. Create elements of `G` with the following (varying) syntax. Pay attention to commas, quotes, brackets, parentheses. The first two use a string (characters) as input, mimicking the way we write permuations (but with commas). The second two use a list of tuples, and so this style might be more useful in programs.

   - `a=G("(5,7,2,9,3,1,8)")`
   - `b=G("(1,3)(4,5)")`
   - `c=G([(1,2),(3,4)])`
   - `d=G([(1,3),(2,5,8),(4,6,7,9,10)])`

3. Compute a^3, bc, $ad^{-1}b$.

4. Compute the orders of each of these four individual elements using a built-in command in Sage.

5. The "sign" of a permutation is ± 1, with value $+1$ for even permutations and -1 for odd permutations. Use this function to determine if a, b, c, d are even or odd permutations.

6. Single elements of a permutation group can be used to generate cyclic subgroups, but you need to input them in a list containing just the one item. Create two cyclic subgroups of G with the commands:

   - `H=G.subgroup([a])`
   - `K=G.subgroup([d])`

   List, and study, the elements of each subgroup. List the size and number of all of the subgroups of K (without using Sage), and construct a subgroup of K of size 10 using Sage.

7. More complicated subgroups can be formed by using more than one generator. Over-simplifying just a bit, imagine forming *all* possible powers, and products of powers of the generators until you achieve a subset that is closed under products and inverses. Construct a subgroup L of G with the command `L=G.subgroup([b,c])`. Compute the order of L and list all of the elements of L.

8. Construct the group of symmetries of the tetrahedron (also the alternating group on 4 symbols, A_4) with the command `A=Alternatinggroup(4)`. Using the tools above, and maybe some Python loops, see if you can find *all of* the subgroups of A_4 (each one exactly once). Provide a nice summary as your answer - not just piles of output. So use Sage as a tool, as needed, but basically your answer will be a concise paragraph and/or table. This is the one part of this assignment without clear, precise directions, so spend some time on this portion to get it right. Hint: no subgroup of A_4 requires more than two generators.

9. Save your work, and then see if you can crash your Sage session with the commands

   - `N=G.subgroup([b,d])`
   - `N.list()`

   How big is N?

Chapter 5

These exercises do not contain much guidance, and get more challenging as they go. They are designed to explore, or confirm, results presented in this chapter. Use some text to block off your work on each of the four questions.

1. Cut/paste the "Subgroup Maker" tool into your worksheet. Use this new Sage function to find an example of a group G and an integer m, so that (a) m divides the order of G, and (b) G has no subgroup of order m. (Do not use the group A_4 for G, since this is in the text.) Provide just enough output to convince the reader that your example is correct.

2. Verify Fermat's Little Theorem (either variant) for your own choice of a single number for the base a, and for every prime number between 100 and 1000. You can experiment as you build your program with lots of output and a smaller range of primes, but the version you submit should only print (a) a prime where the theorem is false, or (b) a message that the theorem is true for all the primes under consideration. (Hint: you might find useful the Python `break` command, and the Sage `prime_range()` command.)

3. Verify that the group of units mod n has order $n - 1$ when n is prime, again for all primes between 100 and 1000. As before, your only output should be a simple message about the truth or falsity of the theorem concerning the order of this group, but you may want to experiment with more copious output (and fewer primes) as you develop your code.

4. Make a copy of your code for (2) and upgrade it to verify Euler's Theorem for every possible $50 \leq n \leq 200$ and $1 \leq a \leq n - 1$ meeting the hypotheses of the theorem. This will require nested loops.

Chapter 8

This exercise is about putting Cayley's Theorem into practice. First, read and study the theorem. Realize this result by itself is primarily of theoretical interest, but with some more theory we could get into some subtler aspects of this (a subject known as "representation theory").
You may discover in this exercise that you are doing a fair amount of pencil-and-paper work, using Sage as a fancy calculator. You do not need to include all these computations in your worksheet. Build the requested group representations and include enough verifications to to provide evidence that your representation is correct.
Begin by building a permutation representation of the quaternions, Q. There are eight elements in Q $(\pm 1, \pm I, \pm J, \pm K)$, so you will be constructing a subgroup of S_8. For each $a \in Q$ form the function λ_a, as defined in the proof of Cayley's theorem. To do this, the two-line version of writing permutations could be useful as an intermediate step. You will probably want to "code" each element of Q with an integer in $\{1, 2, \ldots, 8\}$.
One such representation is in the table at the very end of the Sage Group Theory Primer — your answer should look similar, but perhaps not identical. Do not submit your answer to this, but I strongly suggest doing this particular group until you are sure you have it right — the problems below might be very difficult otherwise.

1. Build a permutation representation of $\mathbb{Z}_2 \times \mathbb{Z}_4$ (remember this group is additive, while the theorem uses multiplicative notation). Then construct the group as a subgroup of a full symmetric group created with two generators. Hint: which two elements of $\mathbb{Z}_2 \times \mathbb{Z}_4$ might you use to generate all of $\mathbb{Z}_2 \times \mathbb{Z}_4$? (Use commands in Sage to investigate various properties of your group, other than just `list()`, to provide evidence that your subgroup is correct — include these in your submitted worksheet.)

2. Build a permutation representation of $U(24)$, the group of units mod 24. Then construct the group as a subgroup of a full symmetric group created with three generators. (Use commands in Sage to investigate various properties of your group, other than just `list()`, to provide evidence that your subgroup is correct — include these in your submitted worksheet.)

Chapter 9a

For each of the groups

1. Alternating group on 4 symbols, A_4
2. The dihedral group of order 8, D_4, the symmetries of a square

do the following:

Use the code I have given you to construct every subgroup. For each subgroup check to see if it is a normal subgroup. You should execute one group of commands once and your output should be a single list where each line is a subgroup and on the same line the words "normal" or "not normal." A good way to check this for a subgroup H is to take each element g from the group, form a left coset and a right coset and test for the equality of these sets. To make equality of these lists happen properly, you want to sort them first. For example, if `left` is a coset stored as a Python list, then `sorted(left)` will be a sorted version of the list.

There is a Sage command you can use as a method on a permutation group, called `G.normal_subgroups()`, to check your work but your program should use just simple Python constructions and no high-powered Sage commands as shortcuts (such as `.is_normal()`.

Finally, for each normal subgroup of A_4 that has order 4, identify generators for the subgroup and construct a "real" Sage subgroup, N, using the `.subgroup()` method of the main group. Now create the quotient group of G by N, using `G.quotient_group(N)`. Explore this new group. Can you explain the output?

Chapter 9b

These exercises are about the variety of possibilities for normal subgroups of a group. The Sage commands are relatively straightforward, but the questions ask for thoughtful responses on your observations.

1. Build a cyclic group of order 40, and a cyclic group of order 41. For each, use the `.normal_subgroups()` command. Give a nicely written explanation of your observations, indicating clearly theorems you might be using.

2. Construct the alternating group on 7 symbols and the full symmetric group on 7 symbols. For each, use the `.normal_subgroups()` command. Comment on your observations.

3. Construct some dihedral groups of order $2n$ (i.e. symmetries of an n-gon, so n will be the parameter you enter into the Sage construction). For each, construct a list of the orders of each of the normal subgroups. Observe enough examples to hypothesize a pattern to your observations and state your hypothesis clearly.

4. Visit the page `http://web.mat.bham.ac.uk/atlas/v2.0/spor/HS/` and scroll down to the line in the "Representation" section which begins "Permutations on 100 points:". At the end of the line, click on the "a" and "b" links for the GAP format of generators. Cut/paste these into your worksheet and use them to create two elements of the symmetric group on 100 symbols. Finally, use these two elements to create a subgroup of S_{100}. This is known as the Higman-Sims group. Use Sage to discover something interesting about this group.

Chapter 11

We now know that every finite abelian group is isomorphic to the direct product of finite cyclic groups. This problem asks you to put this knowledge to work relative to the group of units under multiplication mod n. The general idea is to use the proof of the classification of finite abelian groups (Judson, Theorem 11.3), which is "constructive." This means the proof gives us a recipe to build the object the theorem asserts exists. Once the group is written as an *internal* direct product of finite groups the isomorphism claimed in the theorem is trivial to construct.
You might find useful the commands mentioned in the Chapter 3 exercises. Another useful command is `a.multiplicative_order()` where a is an element of `Integers(n)`.

1. Express the group of units mod 441 as the internal direct product of cyclic groups whose orders are each a power of a prime (i.e. each group will have an order like p^m, but the prime p can differ from one cyclic group to the next). Use Sage's mini-word-processor to write the isomorphism class of the group as an external product of groups $\mathbb{Z}_n$ (`\times` is a useful TEX symbol). As mentioned above, follow the construction in Theorem 11.3, just using Sage as a powerful calculator — do not use high-powered Sage commands that might do this in one or two steps.

2. Repeat the above for the group of units mod 2312.

3. For problem #1, express the element $a = 241$ as a product of powers of generators for your cyclic groups in the internal direct product. (You could probably "brute-force" this with some nested loops that build every possible element of the group as a product of powers of generators until you find the element you want.)

Chapter 12

These exercises will help you build graphs, then ask you to investigate their automorphism groups as group actions on the vertex set. You may use any commands in Sage, some of which you will need to locate yourself — likely using tab-completion on the relevant objects. Here is some help with some relevant Sage commands:
(a) If G is a graph, then `plot(G)` should make a reasonable graphic image of the graph.
(b) Sage may use descriptive labels for a graph's vertices, while automorphism groups use the standard of numbered symbols 1 through n. The syntax
`(A,map) = G.automorphism_group(translation=true)`
will return the automorphism group in `A`, while `map` will translate between the names of the vertices and the integers used in the permutation group. Simply `print map` to see the translation — the syntax should be obvious. Technically, `map` is a Sage "dictionary" if you want to use it programatically.
The requests for commentary here are quite open-ended. There are significant observations to make, so be sure not to be too superficial in your analysis.

1. Build the four-dimensional cube graph, Q_4 with the command `Q = graphs.CubeGraph(4)`. Construct the automorphism group of the graph. Then this group (and any of its subgroups) will provide a group action on the vertex set.

   (a) Construct the orbits of this action, and comment.

   (b) Construct some stabilizers of single vertices (i.e. subgroups of the full automorphism group) and then consider the action of *these* groups on the vertex set. Construct the orbits of these new actions, and comment carefully and fully on your observations, especially in terms of the graph.

2. Build a graph with the following commands:
   `G = graphs.CycleGraph(8)`
   `G.add_edges([(0,2),(1,3),(4,6),(5,7)])`
   The result should be a symmetric looking graph (use `plot()`) with an automorphism group of order 16.

   Repeat parts (a) and (b) from the first exercise.

3. Return to the exercises for Chapter 9 and create the Higman-Sims group. This is an index 2 subgroup of the automorphism group of the Higman-Sims graph (just recently added to Sage, but perhaps not available in the version you are using).

   Repeat parts (a) and (b) from the first exercise, though in this case you will not have the context of the graph.

4. Compare and contrast your observations in the previous three exercises.

5. Build the full symmetric group on 5 symbols and the subgroup A_5 of even permutations. Construct the dihedral groups D_7 and D_8, symmetries of a regular 7-gon and octagon, respectively. For each of these four groups, use a single Sage command to get representatives for the conjugacy classes of the group. Comment on your observations.

Chapter 13

This is an exercise where you will write a short routine mostly using very low-level manipulations of group elements.
Design and create a routine that will accept a permutation group G and a prime number p (which divides the order of the group) and returns a list of all of the Sylow p-subgroups of G. The subgroups returned can just be lists of group elements (in other words, for this exercise, they do not have to be of the proper Sage types to be permutation group objects).
You may use the single high-level Sage command `G.sylow_subgroup()` which will return just one Sylow p-subgroup. Otherwise, avoid any high-powered Sage commands. Use the Second Sylow Theorem (Theorem 13.6, Judson), conjugation, sorted lists, tests of membership in lists, etc to form your list of Sylow p-subgroups (without any duplicates). The dihedral group D_{12}, symmetries of a 12-gon is a reasonable group to use for testing your code.

1. Run your routine for the alternating group A_5 and each possible prime. Confirm the Third Sylow Theorem (Theorem 13.7, Judson) for each prime.

   We know that A_5 is a simple group. Explain how this would explain or predict some aspects of your output.

2. Run your routine for the dihedral group D_{36} (symmetries of a 36-gon) and each possible prime. Confirm the Third Sylow Theorem (Theorem 13.7, Judson) for each prime.

   It can be proved that *any group* of order 72 is not a simple group. Explain how this would explain or predict some aspects of your output.