SuSAAN Exercise Sessions

Aurel Page and Marine Rougnant

5/06/2022 - 14/06/2022

General instructions:

1. Install Pari/GP
2. Do the basic exercises corresponding to each lecture.
3. If you have time, select the advanced exercises or exploration topics that you are interested in.

1 Basic exercises

1.1 Installing Pari/GP

Go to the webpage `http://pari.math.u-bordeaux.fr/Events/COGENT2022`. It also contains advanced instructions here: `http://pari.math.u-bordeaux.fr/Events/COGENT2022/talks/sources.pdf`.

- If you are a Windows user:
  1. Download a precompiled installer by clicking on 32bits (`http://pari.math.u-bordeaux.fr/pub/pari/windows/snapshots/Pari32-2-14-0.G2022.exe`) or 64 bits (`http://pari.math.u-bordeaux.fr/pub/pari/windows/snapshots/Pari64-2-14-0.G2022.exe`).

     Warning: pay attention to the directory in which Pari/GP is installed. If your system refuses to give you access to a particular directory, pick another one. In fact, make sure you have write access to this directory (this is often not the case in Program Files).
  2. Then open the file `gprc` from the pari directory and edit it, including the following lines with modification if needed:
     ```
     breakloop = 1
     parisizemax = "4G"
     \\or the maximum amount of memory that GP can use (important)
     ```
- If you are a Linux user: **do not** install Pari/GP using a package manager (you would get an old version). Use the git version as follows.
  1. Assuming you are using Debian/Ubuntu:
     ```
     sudo apt install build-essential
     sudo apt build-dep pari
     ```

```
sudo apt install libreadline-dev libgmp-dev
sudo apt install git bison automake autoconf
```

2. Then download and compile the source files as follows:
```
git clone https://pari.math.u-bordeaux.fr/git/pari.git
cd pari
./Configure --prefix=GP
make -j4 gp
make docpdf
make bench
make install
make statest-all
```
While the tests are running, you can install optional packages or create the configuration files.

3. You can install optional packages by downloading them from `http://pari.math.u-bordeaux.fr/packages.html` into the pari directory, and then running:
```
make install-data
```

4. Create configuration files: create a file `~/.gprc` and edit it, including the following lines with modification if needed:
```
histfile = "~/.gp_history"
colors = "lightbg" \\or "darkbg"
lines = 40
parisizemax = "4G"
\\or the maximum amount of memory that GP can use (important)
```

5. Run pari with the command `GP/bin/gp`. If you want to be able to run it from any place by simply typing `gp`, run the command:
```
sudo ln -s PATH-TO-YOUR-PARI-DIRECTORY/pari/GP/bin/gp /usr/bin/gp
```
(replacing `PATH-TO-YOUR-PARI-DIRECTORY` by the path to your Pari directory).

- If you are a MacOS user:

  1. Download a precompiled DMG here: `https://pari.math.u-bordeaux.fr/pub/pari/mac/snapshots/PariGP-full-2.14.0.G2022.dmg`. On some systems, you need to go to the file menu and select "Open", so that you can bypass the security check.

     Crash: If Pari/GP starts, prints the banner ("GP/PARI CALCULATOR Version ..."), and then crashes, then

     – Try this one: `https://pari.math.u-bordeaux.fr/pub/pari/mac/snapshots/PariGP-full-2.14.0.G2022-norl.dmg`.
     – If it does not work, you should create a file `.gprc` in your home directory and put the line `readline=0` in this file.

     If the DMG does not work, try the precompiled binary: `https://pari.math.u-bordeaux.fr/pub/pari/mac/snapshots/gp-git-latest-osx`. You may have to make it executable by opening a terminal in the folder containing `gp-git-latest-osx` and typing
     ```
     chmod +x gp-git-latest-osx
     ```

2. Create a file .gprc in your home directory and add the following lines with modification if needed:
```
colors = "lightbg" \\or "darkbg"
lines = 40
parisizemax = "4G"
\\or the maximum amount of memory that GP can use (important)
```

1.2 Introduction to Pari/GP

Exercise 1.2.1 (Documentation)**.**

1. The complete documentation file is the PDF file `doc/users` in the pari installation folder.
2. The `doc` folder also contains short PDF files called `refcard` (with a suffix) that give quick references to many functions of a given topic.
3. In the GP command line, type `?` for a list of help topics.
4. Type `?5` (or another number) for the list of GP functions of a given topic.
5. Short help of a function:
   `?factorial`
6. Long help of a function:
   `??factorial`
   The same help is also contained in the `users` PDF file.
7. Searching for a string in the documentation:
   `???galois`

Exercise 1.2.2 (Basic commands)**.** Play with the following basic commands. You run commands by typing them and then pressing Enter.

1. Arithmetic operations:
```
2+2
4-7
5*7
3^4
4/3
17\3
128%7
```
   Note: `/` is the exact division, \ is the Euclidean division quotient, and `%` is the Euclidean division remainder.
2. Floating-point numbers:
```
2.3
3.9E6
Pi
\p 100
Pi
2^Pi
```

3. Complex numbers:

```
I^2
(1+2*I)*(4-3*I)
(-1)^(1/2)
```

4. Assignment to a variable is `=`, and can be combined with a basic operation:

```
a = 19
b = 11
a*b
a += b
a
```

5. Every non-assigned variable can be used as a polynomial variable:
```
P = x^3 + 13*x + 1
P'
```
The polynomial variable can be recovered with a quote (`'`) **before** the name of the variable:
```
a^2+2*a+3
a = 'a
a^2+2*a+3
```

6. Add a semicolon if you do not want to display the output, or to separate commands on the same line. Compare:

```
N = 2022!
N = 2022!;
N = 2022!; N%2027
```

7. Function call:

```
sqrt(2)
exp(2*I*Pi/3)
abs(3+4*I)
polcoef(P,1)
subst(P,x,1+T)
P
```

Some functions have optional arguments:

```
polcoef((x+1)*(y+2*y^2)*z,1,y)
```

8. Spaces are not significant:

```
sin(10)
s i n ( 1 0 )
```

9. Boolean values are represented by 0 (false) and 1 (true).

```
2 < 3
4 >= 4.
P == 1
a != 3
```

10. Row vectors are created with square brackets, with values (of any type) separated by commas.

    ```
    [1,3,2]
    v = [-1,y,"bla",1/u]
    [-3..10]
    ```

    Their components are numbered from 1 to the length `#v`.

    ```
    v[#v]
    ```

    They can be concatenated with `concat`.

    ```
    concat(v,[1..5])
    concat([[a,b,c],v,[-10..-8]])
    ```

11. Matrices are created with square brackets, with components of a row separated by commas and rows separated by semicolons.

    ```
    M = [1,2;3,4]
    M^2
    ```

12. You can type a multiline command using backslash as follows:

    ```
    N = [7,9;\
         1,8;\
         3,0]
    N*M
    M*N
    ```

13. You can transpose with `~`:

    ```
    v~
    N~
    ```

14. You can access entries of matrices with square brackets:

    ```
    N[3,1]
    N[,2]
    N[,1] = v[2..4]~
    N
    ```

15. You can create vectors and matrices with formulas for the entries:

    ```
    vector(5)
    vector(5,i,i^2+1)
    matrix(3,5,i,j,1/(i+j))
    ```

16. You can write a comment until the end of line with a double backslash:

```
A = 2 \\ +3 this is ignored
A
```

You can write a multiline comment with slash-star:

```
A = 2 + /* this is ignored
and so is this */ 7
```

17. Power series:

```
1/(1+x+O(x^5))
cos(x+O(x^6))
```

18. You can ask for the running time of the last command with `##`, and activate/deactivate the timer with `#`:

```
8^8^8;
##
#
8^9^8;
```

19. Elements of basic quotient rings:

```
Mod(2,7)^3
Mod(x+7,x^4+1)^10
Mod(Mod(3*x+1,x^3+x+1),11)^100
```

You can lift them with `lift` or `centerlift`:

```
lift(Mod(13,7))
lift(Mod(x^7,x^3-x+1))
centerlift(Mod(6,7))
```

20. Type of an object:

```
type(b)
type(a)
type(Pi)
type(v)
type(v~)
type(M)
type(sin)
```

Exercise 1.2.3 (Reading files)**.** Once you start working on exercises requiring more than a few commands (or working on your own projects), you should write your code in a file and read the file in Pari/GP.

1. Create a file `test` with a text editor, and write 10 lines of Pari/GP commands, including some variable assignments. Make sure that there are no spaces in the file name (for instance, **do not** use `my test` as a file name).

   Warning: the lines will be executed one by one, as if you hit Enter between the lines. If you want to write a multiline instruction, you should use backslash, or enclose the block in curly braces. However, you will then need to separate instructions with semicolons, as if they were on the same line.

   ```
   {M = matrix(10,10,i,j,
     a = 1+i+j;
     b = 2+i^2+j^2;
     a/b
   )};
   ```

2. To read the file:

   - If you are a Linux user, type

     `\r test`

     possibly adding a path if the file is not in folder where you started Pari/GP.

   - If you are a Windows user, type

     `\r`

     but do not press Enter. Then, click on the file icon of `test` and drag it to the Pari/GP windows, and drop it on the command line. The complete path to your file should appear. Then, press Enter.

   - If you are a MacOS user, you can use either of the options above.

   Warning: You need to make sure that the file is a plain text file (TXT / `.txt` file), not a file with an enriched format (RTF / `.rtf`, DOC / `.doc`, etc). On MacOS, you can try Format → Make Plain Text.

3. Do you see the output of your commands?

4. What are the values of the variables you that assigned in the file?

5. Add some print statements: `print(something)` to your file (save the changes!).

6. Read it again by simply typing `\r` and Enter. You should see the output of the print statements.

Exercise 1.2.4 (Programming)**.**

1. The syntax to define functions is as follows:

   ```
   f(a,b,c) =
   {
     instruction1;
     instruction2;
   ```

```
  ...
  value_to_be_returned
}
```

You can also return a value in the middle of the function with `return(value)`.

Write a function `f(n)` that returns n^n mod 5. Test it.

2. The syntax of the conditional control structure is as follows:

```
if(condition, instructions_if_true, instructions_if_false)
```

It is better to indent your code as follows:

```
if(condition,
  instruction1_if_true;
  instruction2_if_true;
  etc
,/*else*/
  instruction1_if_false;
  instruction2_if_false;
  etc
)
```

Write a function `parity(n)` that tests the parity of `n`, and prints "even" or "odd" accordingly. Test it.

3. The boolean operators are `&&` (and), `||` (or), `!` (not). Write a function `isgood(n)` that prints "good" when `n` is (not divisible by 10) or (congruent to 1 mod 3 and strictly greater than 30), and "bad" otherwise. Test it.

4. The syntax of the for loops is as follows:

```
for(i=start_value,end_value,
  instruction1;
  instruction2;
  etc
)
```

Write a function `squares(n)` that prints all the squares of integers between 1 and `n`. Test it.

5. The syntax of the while loops is as follows:

```
while(condition,
  instruction1;
  instruction2;
  etc
)
```

Write a function `syracuse(n)` that counts the number of times the following operations must be performed starting from $m = n$ to reach $m = 1$: if m is even, divide it by 2; otherwise, replace it by $3m + 1$. Test it.

6. After running your function `syracuse`, what is the value of `m`? Set `m` to 0, and add the statement (declaration of a local variable)

   ```
   my(m);
   ```

   at the beginning of your function. Test the function again. What is the value of `m` now?

7. Function are allowed to call themselves (recursivity).

   - Write a recursive function `myfact(n)` that computes the factorial of an integer n by using the following properties:
     - $0! = 1$, and
     - $n! = n \cdot (n-1)!$.
   - Write a recursive function `mygcd(a,b)` that computes the GCD of the two integers a, b by using the following properties:
     - $\gcd(a,b) = \gcd(b,a)$,
     - $\gcd(a,0) = a$,
     - $\gcd(2a,2b) = 2\gcd(a,b)$,
     - $\gcd(2a,b) = \gcd(a,b)$ if b is odd, and
     - $\gcd(a,b) = \gcd(a-b,b)$.

     Hint: try to decrease one of (a,b) by doing a substraction or dividing by 2.

     Test your function.

Exercise 1.2.5 (Break loop)**.**

1. Run the following code:

   ```
   {for(i=1,oo,
     for(j=-i,i,
       if(!random(1000), 1/0)
     )
   )}
   ```

2. The code will trigger an error, and GP will end up in a state called the break loop. In the break loop, you can run GP commands, and in particular you can inspect the values of the variables when the error was triggered. Try it with `i` and `j`.

3. You can exit the break loop by tying `break` or pressing `Ctrl+D`.

4. Replace `1/0` by `print(j)`. Run the code again. Press `Ctrl+C`. You enter the break loop again. Here, since there was no error, you can either continue running the code by pressing Enter, or exit the break loop as before.

1.3 Complexity

Exercise 1.3.1 (Bitsize)**.** Recall that a byte is 8 bits. A machine-size integer (a word) is therefore $64/8 = 8$ bytes on a 64 bits machine ($32/8 = 4$ on a 32 bits machine).

1. You can measure the size (in bytes) of any pari object with `sizebyte`. Measure the size of 0, 1, 2, 3, 2^{64}, 2^{128}.
2. After possibly testing more values, predict the general answer for 2^{64k}. Test your hypothesis.

Exercise 1.3.2 (Observing complexities)**.** The goal of this exercise is to get a feeling for different complexity classes.

Here are four functions of an integer variable `N`, of different complexities:

- `f1(N) = N^2` (Fast, quasilinear time)
- `f2(N) = isprime(nextprime(N))` (Medium, polynomial time)
- `f3(N) = factor(randomprime(N)*randomprime(N))` (Slow, subexponential time)
- `f4(N) = primepi(N)` (Very slow, exponential time)

For each of these four functions, do the following.

1. What do these functions do? (use the documentation)
2. You can measure the running time of some instructions in the middle of your code as follows:

   ```
   t = getabstime();
   instruction;
   t = getabstime()-t; \\now t is the time in milliseconds
   print(strtime(t)); \\human-readable format
   ```

   Measure the running time of the function for a few integer values, and observe the dependence on the number of bits of the input.
3. What is (approximately) the maximum bitsize of the input for which the function terminates in less than 1 second?
4. What is (approximately) the maximum bitsize of the input for which the function terminates in less than 2 second?

1.4 Arithmetic operations

Exercise 1.4.1 (Arithmetic operations)**.** Explore and experiment with the following functions (when meaningful, try both integers and polynomials): `fft`, `gcd`, `polresultant`, `znlog`, `factor`.

Exercise 1.4.2 (Fermat's compositeness test)**.**

Let p be an integer. Recall that if p is prime, then for all a coprime to p we have

$$a^{p-1} = 1 \bmod p.$$

1. Implement a function `fermat(a,p)` that tests if this relation is satisfied. Test it. You should be able to run it for `p = nextprime(2^100)`.

2. Implement a function `allfermat(p)` that tests if the relation is satisfied for all a coprime to p.

3. Write code to find a composite integer p such that `allfermat(p)` succeeds.

1.5 Reconstruction

Exercise 1.5.1 (Reconstruction functions)**.** Explore and experiment with the following functions: `round`, `chinese`, `polinterpolate`, `bestappr`, `bestapprPade`.

Exercise 1.5.2 (Cyclotomic polynomials)**.**

- Implement a function `mycyclo(m)` constructing the cyclotomic polynomial $\Phi_m \in \mathbb{Z}[X]$ from the complex roots.

- Compare with `polcyclo`.

1.6 Algebraic number theory

Exercise 1.6.1 (Number fields)**.** Let $Q = x^3 - 111x^2 + 6064x - 189804$.

1. Check that Q is irreducible (`polisirreducible`).

2. Compute a nicer defining polynomial P for the same field (`polredbest`).

3. Check that they really define the same number field (`nfisisom`).

4. Initialise the number field $F = \mathbb{Q}(\alpha)$ defined by P (`nfinit`).

5. What are

   - the signature of F? (`.sign`: usage `F.sign`)
   - the discriminant of F? (`.disc`)
   - a $\mathbb{Z}$-basis of $\mathbb{Z}_F$? (`.zk`)

6. You can represent elements in polynomial form (`Mod(...,P)`) or as column vectors of coefficients on the basis of $\mathbb{Z}_F$. What are the coefficients of $-\frac{5}{2}\alpha^2 + \frac{19}{2}\alpha - 3$ on the basis (`nfalgtobasis`)? Is it an algebraic integer? What are its trace and norm (`nfelttrace`, `nfeltnorm`)?

7. Compute the prime decomposition of $2, 3, 19$ (`idealprimedec`). How many primes ideals are there above them? What are their ramification indices (`.e`)? Residue degree (`.f`)? Compute a basis of these prime ideals (`idealhnf`). Compute the image of some elements in the residue field (`nfmodpr`).

8. Compute a product of some ideals in F (`idealmul`, `idealpow`, `idealfactorback`). Factor it as a product of prime ideals (`idealfactor`). Check the valuations separately (`idealval`).

9. Is F Galois (`galoisinit`)? Does it have automorphisms (`nfgaloisconj`)? What is the Galois group of its Galois closure (`polgalois`)? Compute a defining polynomial of its Galois closure (`nfsplitting`).

Exercise 1.6.2 (Class group and units)**.** To compute the class group and unit group, use `bnfinit`. Let's denote by L the number field defined by $x^3 - x^2 - 54x + 169$
(`L=bnfinit(x^3 - x^2 - 54*x + 169);`).

1. What is L[7]? Find a way to recover it using `L.xxx`.

2. What is the structure of the class group (`.cyc`)?

3. What are the corresponding generators of the class group (`.gen`)?

4. What is the rank of the unit group? What are generators of the unit group (`.tu`, `.fu`)?

5. Explore and experiment with `bnfisprincipal`:

    - Compute the prime decomposition of 13. Let pr be the first component of the output.
    - Express the class of the ideal in terms of the generators of the class group using `bnfisprincipal(L,pr)`. Is this ideal principal?
    - Use `idealfactorback` and `bnfisprincipal(L,pr)` to compute the Hermite normal form of the ideal pr. Compare with `idealhnf(L,pr)`.
    - Show that the square of the ideal pr is a principal ideal.

6. Explore and experiment with `bnfisunit`:

    - Show that the element defined by `u = [0,2,1]~` is a unit of $\mathbb{Z}_L$.
    - Express it in terms of the generators with `bnfisunit`.

Exercise 1.6.3 (Field extensions and subfields)**.**

1. Let $K = \mathbb{Q}[\alpha]$ the field defined by $P = x^4 - x^3 - 3x + 4$. Use `nfinit` to compute K.

2. We consider

$$Q = y^3 + (-\alpha - 1)y^2 + (\alpha^3 + \alpha - 2)y + (-\alpha^3 + 3) \in \mathbb{Q}[\alpha][y].$$

   Check that Q is irreducible over K using `nffactor`.
   Remark: by default, $\mathbb{Q}[x, y] = \mathbb{Q}[y][x]$. To force $\mathbb{Q}[x, y] = \mathbb{Q}[x][y]$, you have to specify `y=varhigher("y")`.

3. Consider the extension $L = K[\beta]$ where β is a root of Q. What is the degree of the extension $L/\mathbb{Q}$?

4. Compute a polynomial which defines $L/\mathbb{Q}$ using `rnfequation`.

5. With `nfsubfields`, find the number of subfields of L. Are some of them isomorphic?

Exercise 1.6.4 (Enumeration of prime ideals)**.** Write a function `nfprimesupto(nf,B)` that computes the list of prime ideals of norm less than B.

Notes: to construct a list of which you do not know the length in advance, you can use `List` and `listput`; you can convert it to a vector afterwards with `Vec`. You can use `forprime` or `primes` to obtain the prime numbers up to a bound.

1.7 Linear algebra and lattices

Exercise 1.7.1 (Matrices)**.**

1. Use the fonctions `matid` and `matdiagonal` to create $A = I_4$ and the diagonal matrix B of size 8 with diagonal coefficients equal to 1,..., 8.

2. With `matconcat`, define:

$$C = \left(\begin{array}{c|c} & a \\ B & \vdots \\ & h \end{array} \right)$$

3. What should be the result of the concatenation of A and B? Try.

4. Define the following bloc matrix:

$$D = \left(\begin{array}{c|c|c} I_4 & \multirow{2}{*}{B} & \multirow{2}{*}{C} \\ \hline I_4 & & \end{array} \right)$$

5. What is the size of D? Use `matsize`.

6. You can also define vectors and matrices with coefficients in the finite field $\mathbb{F}_p$ with `Mod(...,p)`. Write a function `vectors(n,p)` which returns the list of all vectors of $\mathbb{F}_p^n$ (you can use `forvec`).

7. Write a function `columntomatrix(C,n)` that takes as input a vector C of size n^2 and returns the $n \times n$ matrix

$$\begin{pmatrix} C_1 & C_2 & C_3 & \dots & C_n \\ C_{n+1} & C_{n+2} & C_{n+3} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{(n-1)n+1} & \dots & \dots & \dots & C_{n^2} \end{pmatrix}.$$

8. Write a function `matrices(n,p)` which returns the list of all the elements of $\mathcal{M}_n(\mathbb{F}_p)$.

9. Recall that an $n \times n$ matrix M over a field is nilpotent if and only if $M^n = 0$. Write a function `nilpotents(n,p)` that counts (by testing them all) the number of nilpotent matrices in $\mathcal{M}_n(\mathbb{F}_p)$.

Exercise 1.7.2 (Linear algebra over fields)**.**

Let A and Y be the matrices

$$A = \begin{pmatrix} -3 & 1 \\ 6 & -2 \end{pmatrix} \qquad Y = \begin{pmatrix} 9 \\ -18 \end{pmatrix}$$

1. You can compute the determinant of A with `matdet`. Is A invertible?
2. Give a basis of the image of A (`matimage`).
3. Give a basis of the kernel of A (`matker`)
4. Find a solution of the system $AX = Y$ with `matinverseimage`.
5. Deduce the set of the solutions of $AX = Y$.
6. Define an invertible 3×3 matrix A' and a column vector Y'. Solve the system $A'X = Y'$ with both `matsolve` and `matinverseimage`. Is it possible to use `matsolve` in the previous example? Why?

Exercise 1.7.3 (Inverse)**.**
Consider the matrix

$$A = \begin{pmatrix} 1 & 2 & -1 & 4 \\ 7 & 0 & 1 & 3 \\ -1 & 0 & 5 & 1 \end{pmatrix}$$

1. Take you favourite 3×3 invertible matrix. Compute its inverse.
2. Compute the rank of the matrix A. Is A invertible? left-invertible? right-invertible? Try to compute A^{-1}.
3. Same question with ${}^tA$.

Exercise 1.7.4 (Linear algebra over $\mathbb{Z}/N\mathbb{Z}$)**.**
Consider the matrix A of the previouos exercise.

1. Compute a basis of the image of A mod 5 with `matimagemod`. Compare with `matimage(Mod(A,5))`.
2. Compute the kernel of A mod 5 with `matkermod`.
3. Extract from A the matrix formed by the three first columns. Compute its determinant. What should it be mod 5? Check with `matdetmod` and compute the inverse using `matinvmod`. Compare with `Mod(M^-1,5)`. Deduce the solutions of $MX = Y$ mod 5. Find the same result with `matsolvemod`.

Exercise 1.7.5 (Reduction of matrices)**.** Consider the matrix

$$A = \begin{pmatrix} 0 & 4 & 3 \\ 2 & -2 & -3 \\ -2 & 4 & 5 \end{pmatrix}$$

1. Show that $P(X) = X^3 - 3X^2 + 4$ is the characteristic polynomial of A (`charpoly`).
2. Factor P and deduce the eigenvalues of A. Check your result with `mateigen`.
3. Compute the minimum polynomial (`minpoly`). Is A diagonalisable?
4. Give a basis and the dimension of each eigenspace.

5. Give an upper triangular (or diagonal) matrix Δ such that Δ and A are similar.

Exercise 1.7.6 (Operations on vector spaces)**.** Consider the following vectors:

$$u_1 = (0, 1, -2, 1), \quad u_2 = (1, 0, 2, -1), \quad u_3 = (3, 2, 2, -1), \quad u_4 = (0, 0, 1, 0)$$

1. Define these four vectors as column matrices. (`Mat`). Check that it has the correct type with `type`.
2. Give a basis β of the subspace $\mathrm{Vec}\,(u_1, u_2, u_3, u_4)$.
3. Complete β to get a basis of $\mathbb{R}^4$ (`matsupplement`).
4. Compute the intersection of $V_1 = \mathrm{Vec}\,(u_1, u_2)$ and $V_2 = \mathrm{Vec}\,(u_3, u_4)$ (`matintersect`) and check wether it's equal to $\mathrm{Vec}\,(1, 1, 0, 0)$ or not.

1.8 General algorithmic techniques

1.9 L-functions

2 Advanced exercises

2.1 Arithmetic operations

Exercise 2.1.1 (Karatsuba multiplication)**.**

1. Write the coefficients of the product of two degree 1 polynomials
$$(aX + b)(cX + d)$$
2. How many multiplications of coefficients does this formula involve?
3. Expand $(a + b)(c + d)$ and compare it to the coefficient of X. Deduce a formula for the product of the two linear polynomials, that involves only 3 multiplications.
4. Consider the multiplication of two degree $m = 2^k$ polynomials P and Q. Write $P = P_1 X^{m/2} + P_0$ (and similarly for Q). How should we choose the degrees of P_1 and P_0?
5. Deduce a formula for multiplying two degree m polynomials, using 3 multiplications of smaller degree polynomials.
6. Deduce a recursive polynomial multiplication algorithm, and write a pseudocode for it.
7. What is the complexity of your algorithm?
8. Write a function `split(R)` that takes as input a vector representing a polynomial R of even degree and returns a two-components vector, each coefficient being a vector representing R_0 and R_1 as in 4. Test it. (to test your function, see `Vecrev` and `Polrev` to convert between polynomials and vectors of coefficients).

9. Write a recursive function `mult(P,Q)` that takes as input a vector representing two polynomials P, Q of degree a power of 2 and computing their product according to your answer to 6. Test it.

10. Using polynomials with small coefficients and increasing degree, measure the running time of your implementation. Try to predict the running time for a larger instance.

11. For R being each of $P = aX + b$, $Q = cX + d$ and PQ, what are $R(0)$, $R(1)$ and $R(\infty)$ (interpreted as the leading coefficient)? Reinterpret 3 as an evaluation-interpolation algorithm.

Exercise 2.1.2 (Squarefree factorisation)**.** Design and implement an algorithm computing the squarefree factorisation of a polynomial in $\mathbb{Q}[X]$ using GCDs with derivatives. Test it. What is its complexity?

Exercise 2.1.3 (Rabin–Miller compositeness test and square roots mod p)**.**
Let $p > 1$ be an integer, and write $p-1 = 2^t m$ where m is odd (see `valuation`).

Part A (Compositeness test)

1. Let a be coprime to p. If p is prime, prove that either $a^m = 1 \bmod p$ or there exist an i such that $0 \le i < t$ such that $a^{m2^i} = -1 \bmod p$. In the second case, what is $a^{m2^j} \bmod p$ for $j < i$?

2. Implement this compositeness test. Test your implementation.

Part B (Square roots mod p)
Here we assume that p is prime. Let $a \neq 0 \bmod p$.

1. Prove that $a^{(p-1)/2} \bmod p$ is 1 if a is a square in $\mathbb{F}_p$, and -1 otherwise.

2. Implement this test. Test your implementation.

3. Prove that there exists an element g of order 2^t in $\mathbb{F}_p$. What is the probability that r^m has order 2^t, if r is drawn uniformly randomly from $\mathbb{F}_p^\times$?

4. Design and implement a probabilistic algorithm to find g as above. Test it.

5. If $a \in \mathbb{F}_p^\times$ has odd order, find a formula giving a square root of a.

6. If $a \in \mathbb{F}_p^\times$ has order a power of 2, prove that it is a power of g. In terms of this power, give a formula for a square root of p.

7. Design an implement an algorithm writing an element $a \in \mathbb{F}_p^\times$ as a product of an element of odd order and a power of g. Test it.

8. Design an implement an algorithm for computing square roots modulo p. Test it. What is its complexity?

2.2 Reconstruction

2.3 Algebraic number theory

Exercise 2.3.1 (Round 2 for large p)**.** The goal is to implement the Round 2 algorithm to compute a p-maximal order in a degree d field, when $p > d$.

Let $F = \mathbb{Q}(\alpha)$ be defined by $P \in \mathbb{Z}[X]$ monic and irreducible.

Here is a list of polynomials that you can use to test your function, but you should try to produce your own test cases (in particular, you can search number fields in the LMFDB `www.lmfdb.org/NumberField/`).

- $x^2 - 3^{2k+1}$ for various values of k;
- $x^2 - 3 \cdot 5^k$ for various values of k;
- $x^3 - x + 1$;
- $x^3 - 5x^2 - 80x + 25$;
- $x^3 + 27x^2 + 700x + 1875$;
- $x^3 + 200x - 125$;
- $x^4 - 72x^2 + 46$;
- $x^4 - 55x^2 - 25$.

1. Recall how to compute $\mathrm{disc}(\mathbb{Z}[\alpha])$ in terms of P. Write a function `initialdisc(P)` that computes this discriminant.

2. Write a function `initialbasis(P)` that computes a $\mathbb{Z}$-basis of $\mathbb{Z}[\alpha]$, expressed as a vector of polynomials in α.

3. Write a function `traceform(ord)` that takes as input a $\mathbb{Z}$-basis $w_1, \ldots, w_d$ of an order $\mathcal{O}$ (`ord`) and returns the $d \times d$ matrix $(\mathrm{Tr}(w_i w_j))_{i,j}$.

4. Prove that the matrix above is a matrix, in a $\mathbb{Z}$-basis that you will specify, of the map

$$\begin{aligned} \Phi\colon \quad \mathcal{O} &\longrightarrow \quad \mathrm{Hom}(\mathcal{O}, \mathbb{Z}) \\ a &\longmapsto \quad (x \mapsto \mathrm{Tr}(ax)). \end{aligned}$$

5. Let

$$\overline{J_p} = \{a \in \mathcal{O}/p\mathcal{O} \mid \mathrm{Tr}(ax) = 0 \text{ for all } x \in \mathcal{O}/p\mathcal{O}\} = \ker(\Phi \bmod p).$$

Write a function `radicalmod(ord,p)` that takes as input a $\mathbb{Z}$-basis of and order $\mathcal{O}$ and returns an $\mathbb{F}_p$-basis of $\overline{J_p}$.

6. Let

$$J_p = \{a \in \mathcal{O} \mid \mathrm{Tr}(ax) = 0 \bmod p \text{ for all } x \in \mathcal{O}\}.$$

Prove that J_p is the preimage of $\overline{J_p}$ under the map $\mathcal{O} \to \mathcal{O}/p\mathcal{O}$.

7. Write a function `radical(ord,p)` that takes as input a $\mathbb{Z}$-basis of and order $\mathcal{O}$ and returns a $\mathbb{Z}$-basis of J_p (you can use `mathnfmodid`).

8. Write a function `mulmatrix(B,a)` that takes as input a $\mathbb{Q}$-basis B of F and an element $a \in F$, and returns an element of $\mathcal{M}_d(\mathbb{Q})$: the matrix of the map $x \mapsto ax$ in the basis B.

9. Write a function `mulvector(B,a)` with the same input as above, that returns a column vector of size d^2 containing the coefficients of the matrix `mulmatrix(B,a)`.

10. Let
$$\mathcal{O}' = \{a \in F \mid aJ_p \subset J_p\}.$$
Explain why $\mathcal{O}'$ is the set of $a \in F$ such that `mulmatrix(B,a)` has integral entries, where B is a $\mathbb{Z}$-basis of J_p.

11. Write a function `mulorder(B)` that takes as input a $\mathbb{Z}$-basis of J_p and returns a $\mathbb{Z}$-basis of $\mathcal{O}'$ (you can use `matrixqz(...,-2)`).

12. Recall that for $p > d$, an order $\mathcal{O}$ is p-maximal if and only if $\mathcal{O} = \mathcal{O}'$, and that we always have $\mathcal{O} \subset \mathcal{O}'$. Write a function `ispmaximal(ord,p)` that takes as input a $\mathbb{Z}$-basis `ord` of an order $\mathcal{O}$ and a prime $p > d$, and returns 1 if $\mathcal{O}$ is p-maximal, and a $\mathbb{Z}$-basis of a strictly larger order otherwise.

13. Write a function `pmaximalorder(P,p)` that takes as input a defining polynomial P and returns a $\mathbb{Z}$-basis of a p-maximal order in $\mathbb{Z}_F$.

14. Write a function `pdiscval(P,p)` with the same input as above, that returns the exponent of p in the discriminant of the field F.

2.4 Linear algebra and lattices

2.5 General algorithmic techniques

2.6 L-functions

3 Exploration

3.1 Arithmetic operations

Exercise 3.1.1 (Divisibility properties of integers).

1. Let $y > 1$. Consider the class of integers N such that all prime factors of N are greater than y. Experiment with the proportion of such integers. Can you conjecture a value? Prove it? (useful advanced loop: `forfactored`)

2. Let $y > 1$. Consider the class of integers N such that all prime factors of N are less than y. Experiment with the proportion of such integers. What does the proportion look like? For large values, it may be better to observe the proportion by taking random large integers instead of all up to a bound.

3. Considering that we can find small prime factors more easily than a complete factorisation, what is the shape of easily factorable numbers? Experiment with the options of `factorint` (the algorithm mentioned in the lecture to find small prime factors is called ECM). What does the proportion of easily factorable numbers look like?

3.2 Reconstruction

3.3 Algebraic number theory

Exercise 3.3.1 (Statistics on class group)**.** The general question is the following: Let Ab be the set of isomorphism classes of finite groups. Let $f\colon \mathrm{Ab} \to \mathbb{R}$ be a map, and let $\mathcal{F}$ be a family of number fields (for instance: all quadratic fields). For $X > 0$, let $\mathcal{F}_X$ be the set of elements of $\mathcal{F}$ of discriminant up to X. We are interested in the behaviour when X tends to infinity, and in particular in the limit if it exists, of the quantity:

$$E_{\mathcal{F},X}(f) = \frac{\sum_{F\in\mathcal{F}_X} f(\mathrm{Cl}(F))}{|\mathcal{F}_X|}.$$

If the limit exists, we write it $E_{\mathcal{F}}(f)$ and we call it the average of f in the family $\mathcal{F}$. If f is the indicator function of a subset Y of Ab, we call it the probability that the class group belongs to Y in the family $\mathcal{F}$.

1. Write a function that enumerate quadratic fields by increasing discriminant and computes their class groups.
2. Write a function that computes $E_{\mathcal{F},X}(f)$.
3. For some maps f of your choice, does $E_{\mathcal{F},X}(f)$ seem to approach a limit? For instance, $f(A) = 1$ if A is the trivial group and $f(A) = 0$ otherwise, or $f(A) =$ the p-rank of A for a fixed prime p (`snfrank`).
4. Does the limit stay the same in subfamilies? For instance, does it depend on the signature of the fields? On the number of ramified primes? On the decomposition type of small primes?
5. Does the probability that a prime number p divide the class number look like $1/p$, as it would for a random integer?
6. What happens if you look at the family of quadratic fields defined by a polynomial $x^2 - tx + 1$, ordered by increasing $|t|$? Are these fields real or imaginary?
7. What happens in higher degrees? Is it influenced by the Galois group? You can generate fields with `nflist` or download fields from `www.lmfdb.org/NumberField/`.

3.4 L-functions