Tartalomjegyzék

Using Sage

Sage server

https://sage.math.bme.hu/

If your browser finds the certificate untrustworthy, accept it manually!

More matrices

Reminder and new stuff

In sage we can define a matrix as follows:

```
m = matrix([[1, 0], [0, 1]])
```

This results in the matrix:

```
1 0
0 1
```

Blockmatrices, all one matrices, diagonal matrices can be created easily, like in octave:

```
A = diagonal_matrix([1, 5])
B = ones_matrix(2, 2)
block_matrix([[A, -1*A], [A^(-1), B]])
```

This results in the matrix:

```
 1   0| -1   0
 0   5|  0  -5
------+-------
 1   0|  1   1
 0 1/5|  1   1
```

The **det** method calculates the determinant of the matrix:

```
m.det()
```

Tasks

Blockmatrix

Calculate the determinant of the following blockmatrix:

```
X I
O X
```

where I is the 3x3 identity matrix and O is the 3x3 all 0 matrix, X is the following:

```
 0 -1 -1
-1  0 -1
-1 -1  0
```

Equations

Solve the following system of equations with the form Ax = b, where A and b are:

```
 1 -1  0  |  1
 3  1 -1  |  1
-2  0  1  |  2
```

Use the **solve_right** method from the lecture!

Once you have the solution, make the matrix into a matrix over the ring GF(3) (with the **change_ring** method), solve it with this new matrix.

Linear independence

Find out for which values of **x** would the rows (or columns) of the following matrix be dependent / independent. (Use the **solve** method.)

```
x  0  1
0  2  x
1  x -1
```

List comprehension

Reminder

```
[expression for element in iterable_thing]
```

This creates a list which contains the **expression** for every element of **iterable_thing**. An iterable thing is a list for example, like a list created with the **range** function.

```
[expression if condition for element in iterable_thing]
```

Similar to the previous one, except only the elements for which the **condition** holds will be included.

```
[expression if condition1 else expression_alt for element in iterable_thing1
           for element2 in iterable_thing2
           ...
           for elementN in iterable_thingN]
```

We can write multiple fors.

Example:

```
[n^2 for n in range(1, 5)]  # [1, 4, 9, 16]
[n for n in [-1, 2, -3, 4] if n > 0]  # [2, 4]
```

Tasks

What do these do?

Execute the following, and then study how they work.

```
[n for n in range(1, 10)]

[(n, m) for n in range(1, 10) for m in range(1, 5)]

[n for n in range(1, 10) if is_prime(n)]

[n for n in range(1, 100) if n % 5 == 0 and n % 7 == 1]

[(n, m) for n in range(1, 5) for m in range(n, 5)]

[(m, n) for n in range(1, 10) for m in range(n, 10) if m % n == 0]

sorted([(m, n) for n in range(1, 10) for m in range(n, 10) if m % n == 0])

sum([n for n in range(1, 10) if is_prime(n)])
```

A little spoiler for the following: spoiler

```
[n for n in range(1, 100) if n == sum([m for m in range(1, n) if n % m == 0])]
```

Solve the following

1. Find the square numbers x below 1000, where x + 1 is a prime. For example 4. (Find all of these numbers.)
2. Find those (x, y) pairs of numbers, where both are prime and their integer division (//) is prime as well. For example (11, 2).
3. Find the 3 digit numbers with the form xyz, where xyz (= 100 * x + 10 * y + z) = x^3 + y^3 + z^3. For example 1, 5, 3, because 1^3 + 5^3 + 3^3 == 153.
4. Find the numbers below 10000, which can be written as the sum of 2 cube numbers in two different ways.