„Równania liniowe” – zadania kwalifikacyjne

Zamieszczanie rozwiązań

Rozwiązania należy wysłać poprzez aplikację warsztatów (warsztatywww.pl). Możesz je wrzucić jako osobne pliki, albo spakowane w archiwum, jak ci wygodniej. Proszę jednak, abyś pliki tekstowe zapisał w kodowaniu UTF-8 oraz używając Unixowych zakończeń linii (LF).

Kontakt (Jakub Nowak)

Wszelkie pytania, przemyślenia, niejasności i inne zmartwienia skieruj pod dowolny z poniższych kanałów:

Email: j.nowak26+warsztatywww@student.uw.edu.pl

Discord: MrQubo#2852

Zadanie 0 [0 pkt.] Napisz coś o sobie. Co cię interesuję, czym się zajmujesz, możesz pochwalić się jakimś swoim projektem. Nie musisz pisać rozprawki, wystarczą 2–3 zdania. Forma rozwiązania: dowolna.

Algebra liniowa

Zadanie 1.1 [2 pkt.] Rozwiąż poniższe układy równań (tzn. podaj wszystkie rozwiązania). Wystarczy podać same rozwiązania, bez obliczeń.

$$\begin{cases} x + y + z & = 1 \\ x + 2y + 4z & = 8 \\ x + 3y + 9z & = 27 \end{cases}$$

$$\begin{cases} x + y + z & = 1 \\ x + 2y + 4z & = 8 \\ 2x + 3y + 5z & = 9 \end{cases}$$

$$\begin{cases} x + y + z & = 1 \\ x + 2y + 4z & = 8 \\ 3x + 4y + 6z & = 9 \end{cases}$$

Zadanie 1.2 [2 pkt.] Rozwiąż poniższy układ równań przy pomocy macierzy i metody eliminacji Gaussa (znaną również jako triangulacja macierzy). Obliczenia zapisz na kartce i zamieść skan, bądź zdjęcie. Można również zapisać obliczenia w LaTeX-u i wysłać plik pdf.

$$\begin{cases} x + y + z & = 1 \\ x + 2y + 4z & = 8 \\ x + 3y + 9z & = 27 \end{cases}$$

SageMath

Trzeba zainstalować SageMath na swoim komputerze. Najlepiej skorzystać z managera pakietów. W razie problemów, proszę o kontakt, postaram się pomóc.

Spróbuj uruchomić konsolę interaktywną poleceniem `sage`. W celu przetestowania czy wszystko działa można przekopiować poniższe dwie linijki:

```
R.<x> = QQ[]
factor(x^2 - 1)
```

Warto sobie również ogarnąć edytor tekstu do Sage. Nie trzeba mieć specjalnego wsparcia dla składni Sage, ten język jest bardzo podobny do Pythona. Wystarczy ustawić język składni na Pythona, aby mieć dobre kolorowanie. Można również używać Jupytera.

Sage jest w zasadzie lekko zmodyfikowanym Pythonem. Ważna różnica o której warto pamiętać, to operator `^` w Sage to potęgowanie (`**` w Pythonie), a nie xor. Operator xor w Sage to `^^`.

Jako rozwiązania poniższych zadań wyślij kody źródłowe. Można to zrobić na dwa sposoby, albo jako pliki tekstowe z rozszerzeniem `.sage`, albo jako notebooki Jupytera (z rozszerzeniem `.ipynb`).

Liczby modulo

Będziemy korzystać z arytmetyki modularnej. Jej implementacja w Sage jest bardzo wygodna i pozwala nam na zapisywanie wzorów podobnie jakbyśmy zapisali je na kartce.

Aby dodać do siebie dwie liczby `a` i `b` modulo 37 w Pythonie, najprawdopodbniej zrobilibyśmy to tak: `(a + b) % 37`. Sage pozwala nam powiedzieć, że `a` i `b` są „liczbami modulo 37" i wszystkie operacje na tych liczbach należy wykonywać modulo 37.

Spróbuj wpisać `type(2)` w Sage. Dowiesz się, że liczba 2 jest typu `Integer`, innymi słowy, jest to po prostu liczba całkowita. Aby stworzyć zmienne modulo, musimy najpierw stworzyć typ dla zmiennych modulo. W tym celu należy użyć konstruktora `Zmod`. `Zmod(37)` zwraca typ zmiennych modulo 37. Możemy zapisać ten typ do zmiennej `R`: `R = Zmod(37)`. Następnie możemy stworzyć zmienną tego typu: `a = R(2)`. Spróbuj teraz wpisać `type(a)`. Dowiesz się, że `a` jet typu `IntegerMod`. Sprawdź, jaki wynik otrzymasz gdy wpiszesz np. `a^10`. Porównaj wynik z `(2^10) % 37`.

NB: Formalnie powinienem powiedzieć, że `Zmod(37)` jest pierścieniem (a nawet ciałem) liczb modulo 37, ale nie trzeba wiedzieć, czym jest pierścień.

Zadanie 2.1 [2 pkt.] Oblicz (w Sage) i wypisz $3^{13^{37}} \pmod{101}$.

Powinno wyjść 61.

Macierze

Wpisując `matrix?` w interaktywnej konsoli Sage możesz zaznajomić się z konstruktorem macierzy.

Zadanie 2.2 [1 pkt.] Skonstruuj w Sage poniższą macierz:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

Wypisz tę macierz podniesioną do kwadratu.

Macierze modulo Konstruktor macierzy przyjmuje również opcjonalny argument `"ring"`. Możemy jako ten argument przekazać nasz typ liczb modulo.

Zadanie 2.3 [2 pkt.] Skonstruuj w Sage poniższą macierz:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

Wypisz tę macierz podniesioną do potęgi $10^9 + 7 \pmod{101}$.

Powinno wyjść

$$\begin{bmatrix} 87 & 99 & 5 \\ 62 & 92 & 23 \\ 42 & 8 & 96 \end{bmatrix}$$

Układy równań

Układy równań liniowych można rozwiązywać w Sage przy pomocy macierzy i metody `.solve_right()`. Przykład tutaj: https://doc.sagemath.org/html/en/tutorial/tour_linalg.html

Zadanie 2.4 [1 pkt.] Rozwiąż poniższy układy równań przy pomocy macierzy, w Sage:

$$\begin{cases} x + y + z & = 1 \\ x + 2y + 4z & = 8 \\ x + 3y + 9z & = 27 \end{cases}$$

Zadanie 2.5 [7 pkt.] Rozwiąż poniższy układy równań przy pomocy macierzy, w Sage:

$$\left\{\sum_{j=0}^{N-1} i^j x_j = i^N \quad (\text{for } i = 1, \ldots, N)\right.$$

Dla jasności: Jest to układ N równań o N niewiadomych. Jako N możesz podstawić np. 3, ale napisz kod w taki sposób, aby N dało się łatwo, dowolnie zmienić. Nie chodzi w tym zadaniu o podanie rozwiązania dla wszystkich możliwych N, a o napisanie kodu, który rozwiąże ten układ dla dowolnego, zadanego N.

Dla N = 42 wyszło mi

```
(-14050061177528798985431426062445115699363840000000000,
607910011328418199588537733118255601985519616000000,
-120123088506928130571484361778312932677826641920000000,
146019971727550114046899451463617794649666617344000000,
-123718341827855166687835365459107945846831107276800000,
78335002753989806506258209591143967192818649661440000,
-3875872186584600545355596056500496761371564821708800,
154584698940834404355578835332730709702318453948416 0,
-508461669133944620833708014644828000113268994342912,
140348823241521672911308014781474647177596666585088,
-32957450733681922322799734591224832813970540374016,
665650937252724453842471970507906713323770447360 0,
-116668257596995758491813343216135384271586187110 4,
178750894271463502156317018773923052439354713856,
-24085776349125166346035563126897482872127590912,
28686015980633221454962704443699786487361555 20,
-30323840467329050597197885376498775068016313 6,
28549067543671985479632160414852425937484544,
-24005244403842215038101448817992818453617 68,
18067309449021182753210996658646237133982 0,
-12192616509244184667134003029133021482178,
73867124853752332859675949132996204920 7,
-40206174229950065906142659931071804669,
196681858099600430006649313080590715 0, -864598688440861812080872004918308 40,
34134452510007037144865921647894 35, -12090343842043571697403688491204 5,
38358685751589902098447322738 40, -10877478115734078687618778042 8,
27491973169130021099899634 22, -61707382118599937573753754,
12245152489601361446025 00, -21362021444009549530136, 325322751495206555454,
-4286778229133894858, 48326290119924180, -459295484924354, 3608591714091,
-22808599177, 111439230, -394912, 903)
```

Zadanka z crypto

W poniższych zadaniach celem jest znalezienie flagi, tzn. ciągu znaków, w formacie „CRY{[a-z-]}”, np. „CRY{fajna-flaga}”.

Te zadania mogą być trochę trudniejsze, zrobienie wszystkich poprzednich wystarczy do kwalifikacji.

Można rozwiązać jak i w czym się chce. Proszę o załączenie skryptów i opisu rozwiązania (nie musi być długi).

Uzywany moduł „Crypto” można zainstalować przez pip. Nie jest to konieczne do rozwiązania zadania, ale może pomóc: https://pypi.org/project/pycryptodome/

Zadanie 3.1 [3 pkt.] http://3.120.130.198/challenges#Quals%201-1

Zadanie 3.2 [2 pkt.] http://3.120.130.198/challenges#Quals%202-2

Zadanie 3.3 [5 pkt.] http://3.120.130.198/challenges#Quals%203-3

bytes_to_long `Crypto.Util.number.bytes_to_long()` konwertuje bajty na liczbę. Konwersję w druga stronę można zrobić przy pomocy `Crypto.Util.number.long_to_bytes()` albo funkcji pack z pakietu pwntools (trzeba przekazać argument `"all"`).

Kontakt (Jakub Nowak)

Email: j.nowak26+warsztatywww@student.uw.edu.pl

Discord: MrQubo#2852