Sage - Niezbędne Komendy na MO

Rozdział 1

Komendy z Zestawów

1.1 Zestaw 1

- **Deklarowanie zmiennej**

  ```
  var('x')
  ```

  Zadeklarowanie zmiennej x

- **Ładne wypisywanie wzorów**

  ```
  show(f)
  ```

  Wypisanie postaci wyrażenia f w postaci estetyczniejszej i wyśrodkowanej.

- **Rozkład liczby na czynniki pierwsze**

  ```
  factor(2^(13)-3^(12))
  ```

  Rozłożenie liczby $2^{13} - 3^{12}$ na czynniki pierwsze.

- **Konstrukcja ciała $\mathbb{Z}_n$**

  ```
  K = GF(13)
  ```

  Przypisanie zmiennej K ciała $\mathbb{Z}_{13}$.

- **Wyliczenie wartości danego wyrażenia w ciele K**

  ```
  K(2^1234554321)
  ```

  Wyliczenie wartości wyrażenia $2^{1234554321}$ w ciele oznaczonym jako$K$.

- **Konstrukcja macierzy**

  ```
  A = matrix(K, [[1, 3, 5, 7], [2, 3, 1, 8], [1, 1, 0, 9]])
  ```

Konstrukcja macierzy $\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 3 & 1 & 8 \\ 1 & 1 & 0 & 9 \end{bmatrix}$ o współczynnikach z ciała K.

- **Zadeklarowanie pierścienia wielomianów**

```
P.<x> = K[]
```

Deklaracja pierścienia wielomianów zmiennej x nad ciałem K

- **Macierz jednostkowa**

```
I = identity_matrix(K,4)
```

Macierz jednostkowa o wymiarach 4×4 nad ciałem K

- **Konstrukcja listy z podanymi ręcznie wartościami**

```
list1 = [1,7,3,15]
```

Budowa listy $[1, 7, 3, 15]$

- **Konstrukcja listy o danym początku i końcu i skoku** 1

```
list2 = [2,..,16]
```

Budowa listy $[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]$

- **Konstrukcja listy o danym początku i końcu i określonym skoku**

```
list3 = [2,4..,16]
```

Budowa listy $[2, 4, 6, 8, 10, 12, 14, 16]$ o skoku równym $4 - 2 = 2$ (różnicy pierwszych wartości z listy).

- **Pętla FOR o liczniku z listy**

```
L = [2^j for j in[1..10]]
```

Pętla **for** działająca od 1 do 10 (czyli na wartościcach z listy $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$)

- **Pętla FOR o liczniku z określonego zbioru danych z dołączonym warunkiem**

```
L2 = [2^p-1 for p in prime_range(700) if (2^p-1).is_prime()]
```

Pętla **for** o licznikach będących liczbami pierwszymi, które dodatkowo spełniają warunek, że $2^p - 1$ też jest liczbą pierwszą.

- **Konkatenacja list**

```
M4 = M1+M2+M3
```

Lista M4 jest sumą list M1,M2,M3.

- **Wypisanie określonego wyrazu listy**

```
L[j]
```

Wypisanie wartości j-tego miejsca z listy L (UWAGA: Listy numerowane są 0).

1.2 Zestaw 2

- **Konstrukcja funkcji i jej wywołanie**

```
def potegi(n):
    L = [2^j for j in[0..n]]
    show(L)
potegi(4)
```

Funkcja wypisująca listę wszystkich potęg liczby 2 o wykładnikach z zakresu $[0, n]$ oraz jej wywołanie dla argumentu 4.

- **Określanie długości listy**

```
dl = len(L)
```

Zapisanie do zmiennej *dl* długości listy L.

- **Instrukcja warunkowa IF**

```
if n == 2:
    n=n+1
```

Jeśli liczba n jest równa 2 to dodaj do niej wartość 1.

- **Pętla FOR o liczniku z danego zakresu o skoku** 1

```
for i in range(1,  n)
```

Pętla **for** działa od licznika równego 1 do $n - 1$ ze skokiem 1.

- **Najwyższy współczynnik wielomianu**

```
f.leading_coefficient()
```

Wypisanie najwyższego współczynnika wielomianu f.

- **Stopień wielomianu**

```
f.degree()
```

Wypisanie stopnia wielomianu.

- **Macierz zerowa nad ciałem identycznym jak ciało wielomianu**

```
D = zero_matrix(f.base_ring(),d)
```

Macierz D będzie kwadratową macierzą zerową o wymiarze d nad ciałem współczynników identycznym jak wielomian f.

- **Normowanie wielomianu**

```
g = f.monic()
```

Wielomian g jest wielomianem powstałym poprzez unormowanie wielomianu f.

- **Liczba wierszy i kolumn macierzy**

```
M.nrows()
M.ncolumns()
```

Wypisanie ilości wierszy i kolumn macierzy M.

- **Transpozycja macierzy**

```
M.transpose()
```

Transponowanie macierzy M.

- **Sprawdzenie czy macierz M jest kwadratowa**

```
if M.is_square():
```

Sprawdzenie czy M jest macierzą kwadratową.

- **Dołączenie elementu do listy**

```
S.append(a)
```

Dołączenie elementu a do listy S

- **Usunięcie elementu z listy**

```
S.pop(i)
```

Usunięcie z listy S wartości z i-tego miejsca.

- **Konwersja znaku na liczbę zmiennoprzecinkową**

```
float(a)
```

Konwersja znaku a na liczbę.

- **Pętla WHILE**

```
i = 1
while (i < len(L)):
      n=n+i
      i=i+1
```

Dopóki i jest mniejsze od długości listy L to dodawaj za każdym razem i do n i zwiększaj i o 1.

1.3 Zestaw 3

- **Funkcja określona kilkoma wzorami**

```
f = piecewise([[(-pi,0),-1],[(0,pi),1]])
```

Funkcja przyjmuje wartość -1 na przedziale $(-\pi, 0)$ i wartość 1 na przedziale $(0, \pi)$.

- **Wykres funkcji na przedziale**

```
plot(f,(x,-pi,pi))
```

Wykres funkcji f o argumencie x na przedziale $(-\pi, \pi)$

- **Całkowanie**

```
w = integral(-cos(2*n*pi*x/L),x,-L/2,0)
```

Całka z funkcji $-\cos(nx\frac{2\pi}{L})$ względem zmiennej x na przedziale $(-\frac{L}{2}, 0)$

- **Dostosowywanie wykresu funkcji**

```
F.plot(-10*pi,10*pi,aspect_ratio = 2)
```

Wykres funkcji F na przedziale $(-10\pi, 10, \pi)$ zostanie dostosowany do lepszego wyświetlania w zależności od otrzymanych wartości funkcji na tym przedziale.

- **Uproszczenie wyrażenia algebricznego**

```
F(T).simplify_full()
```

Wyrażenie F o argumencie T zostanie uproszczone do jak najkrótszej postaci.

- **Przybliżona, numeryczna wartość liczby zespolonej**

```
CC((cos(2*pi/n)+i*sin(2*pi/n)))
```

Liczba $\cos(\frac{2\pi}{n}) + i\sin(\frac{2\pi}{n})$ zostanie zapisana w postaci ze współczynnikami dziesiętnymi zamiast współczynników pierwiatkowych.

1.4 Zestaw 4

- **Warunek niezbędne do działania funkcji**

```
def iloczyn_skalarny( U, V ) :
       assert len(U) == len(V), "Listy U, V muszą mieć równą długość!"
```

Funkcja odpali się tylko, gdy listy U i V będa takiej samej długości, inaczej nastąpi przerwanie i pojawi się komunikat o błędzie "Listy U, V muszą mieć równą długość!".

- **Macierz o losowych współczynnikach**

```
M = random_matrix(ZZ,5,7)
```

M będzie macierzą o losowych współczynnikach całkowitych.

- **Wypisywanie czasu działania danej funkcji**

```
%timeit
iloczyn_skalarny([2,3,2],[1,4,2])
```

Wypisanie czasu działania funkcji `iloczyn_skalarny`

- **Grupa permutacji**

```
SymmetricGroup(3)
```

Wypisanie grupy permutacji zbioru 3-elementowego, gdy użyjemy wraz z komendą `show` wypisze nam zbiór generatorów tej grupy permutacji.

- **Znaki permutacji**

```
sn = SymmetricGroup(3)
for sigma in sn:
    sigma.sign()
```

Wypisanie wszystkich znaków permutacji zbioru 3-elementowego.

- **Wycięcie części macierzy**

```
A.submatrix(0,0,2,3)
```

Wycięcie z macierzy A fragmentu do drugiego wiersza i do trzeciej kolumny.

- **Scalanie macierzy**

```
block_matrix(A,B)
```

Scalenie macierzy A i B w nową macierz.

- **Macierz diagonalna**

  ```
  M.diagonal()
  ```

  Wypisanie przekątnej macierzy M.

- **Wypisanie współczynników w macierzy od pewnego miejsca**

  ```
  A.diagonal()[2:]
  ```

  Wypisanie przekątnej macierzy M, ale od miejsca 2.

- **Tworzenie kopii**

  ```
  A = copy(M)
  ```

  A jest kopią elementu np. macierzy M.

1.5 Zestaw 5

- **Największy wspólny dzielnik**

  ```
  gcd(35,45)
  ```

  Wypisania największego wspólnego dzielnika liczb 35 i 45, komenda ta działa również na elementach pierścienia wielomianów.

- **Przechwytywanie pierścienia z elementu**

  ```
  P = f.parent()
  ```

  Oznaczenie poprzez P pierścienia, nad którym określony jest element f.

- **Pochodna**

  ```
  f.derivative()
  ```

  Wyliczenie pochodnej funkcji f.

- **Rozszerzony algorytm Euklidesa**

  ```
  xgcd(35,45)
  ```

  Wyliczenie NWD elementów 35 i 45 oraz wypisanie współczynników α i β z równania diofrantycznego $35\alpha + 45\beta = NWD(35, 45)$

- **Macierz stowarzyszona**

```
companion_matrix(r)
```

Wypisanie macierzy stowarzyszonej z wielomianem r.

- **Tworzenie wektora**

```
vector ([j for j in range (10)])
```

Stworzenie wektora składającego się z kolejnych dziesięciu liczb naturalnych.

- **Tworzenie wektora na bazie współczynników wielomianu**

```
vector(g.list())
```

Tworzenie wektora na bazie współczynników wielomianu g.

- **Iloczyn wartości z listy podniesionych do odpowiednich potęg**

```
Factorization(g, unit = lcf)
```

Faktoryzacja listy list g (czynnik, potęga) względem współczynnika (lcf)

- **IF uwzględniające wartości stałe**

```
if g.is_constant():
```

Sprawdzanie czy wielomian g jest stałą czy też nie.

1.6 Zestaw 6

- **Licznik ułamka**

```
r.numerator()
```

Wypisanie licznika ułamka r.

- **Mianownik ułamka**

```
r.denominator()
```

Wypisania mianownika ułamka r.

- **Wypisania części całkowitej i reszty z dzielenia**

```
f.quo_rem(g)
```

Wypisanie części całkowitej i reszty z dzielenia elementu f przez element g.

- **Wyciągnięcie pojedynczego elementu z listy list**

```
G[0][1]
```

Wyciągniecie drugiego elementu z pierwszej podlisty w liście G

Rozdział 2

Ogólna Baza Komend

2.1 Podstawy

Komenda	Opis
`var('x')`	Deklarowanie zmiennej x
`show(f)`	Ładne, wyśrodkowane wypisanie wyrażenia f

2.2 Macierze

`M = matrix([[1,2,3],[4,5,6],[7,8,9]])`	Podstawowa konstrukcja macierzy
`M = matrix(CC,[[1,2,3],[4,5,6],[7,8,9]])`	Konstrukcja macierzy nad odpowiednim ciałem np. liczb zespolonych.
`M.det()`	Wyznacznik macierzy M
`M.trace()`	Ślad macierzy M
`M.transpose()`	Transpozycja macierzy M
`M.inverse()`	Macierz odwrotna do macierzy M
`I = identity_matrix(K,n)`	Deklaracja macierzy jednostkowej I o współczynnikach z ciała/pierścienia K i wymiarach $n \times n$.

2.3 Pierścienie i ciała

`K = GF(13)`	Konstrukcja ciała skończonego $\mathbb{Z}_{13}$ i oznaczenie go jako K.
`K(a)`	Wyliczenie wartości elementu a w pierścieniu/ciele K.
`P.<x> = K[]`	Tworzenie pierścienia P zmiennej x nad ciałem/pierścieniem K, ($K = QQ$ - wymierne, RR - rzeczywiste, CC - zespolone, ZZ - całkowite).

2.4 Faktoryzacje i rozkłady

`factor(a)`	Rozkład liczby a na czynniki pierwsze

2.5 Listy

`L = [1,9,30,100]`	Stworzenie listy złożonej z elementów $1, 9, 30, 100$.
`L = [2,..,16]`	Stworzenie listy złożonej z kolejnych liczb od 2 do 16.
`L = [2,4..,16]`	Stworzenie listy złożonej z liczb o kroku określonym przez różnicę dwóch pierwszych liczb listy.
`L = [2^j for j in[1..10]]`	Generowanie listy przy użyciu pętli
`L = [2^p-1 for p in prime_range(700) if (2^p-1).is_prime()]`	Generowanie listy przy użyciu pętli z dołączonym warunkiem.
`prime_range(1000)`	Wypisanie listy liczb pierwszych z określonego zakresu.

2.6 Warunki

`a.is_prime()`	Zwrócenie wartości "True", gdy liczba jest pierwsza lub "False", gdy tak nie jest.