Implementacja wielomianów na bazie tablic

Przemysław Pawlik

1. Opis programu

Reprezentacja wielomianów na bazie tablic jest jedną z najczęściej używanych reprezentacji do przechowywania i operowania na wielomianach. Wielomiany przedstawiane i przechowywane są w postaci od najmniejszej potęgi x.
Działanie na samych wielomianach sprowadza się do kilku prostych instrukcji:

1. Tworzenie wielomianu o zadanych współczynnikach
2. Dodawanie dwóch wielomianów
3. Odejmowanie dwóch wielomianów
4. Mnożenie dwóch wielomianów
5. Obliczanie wartości wielomianu dla danego argumentu

---

2. Implementacja

Aplikacja składa się z jednej klasy: `Wielomian` - zajmującej się przechowywaniem i operacjami na wielomianach.

Tworzenie wielomianu

```
class Wielomian:
    """
    Klasa reprezentujaca wialomian
    """
    def __init__(self, wspolczynniki):
        """
        Inicjalizacja wielomianu o zadanych współczynnikach
        """
        stopien = self.__stopien(wspolczynniki)
        self.wspolczynniki = [None] * stopien

        for i in range(stopien):
            self.wspolczynniki[i] = float(wspolczynniki[i])
```

Na początku tworzenia wielomianu inicjalizowana jest nowa tablica o wielkości odpowiadającej stopniowi tworzonego wielomianu. Następnie do tak przygotwanej tablicy kopiowane są współczynniki wielomianu.

Dodawanie wielomianów

```
def __add__(self, other):
    """
    Funkcja dodaje do siebie dwa wielomiany
    """
    wspolczynniki = None
    stopien1 = self.__stopien(self.wspolczynniki)
    stopien2 = self.__stopien(other.wspolczynniki)
```

Żeby w łatwy sposób dodawać wielomiany przeciążany został operator +. Zanim program przystępuje do dodawania pobiera stopnie obu wielomianów.

```
    if stopien1 > stopien2:
        wspolczynniki = [None] * stopien1
        for i in range(stopien2):
            wspolczynniki[i] = self.wspolczynniki[i] + other.wspolczynniki[i]

        for i in range(stopien2, stopien1):
            wspolczynniki[i] = self.wspolczynniki[i]

    else:
        wspolczynniki = [None] * stopien2
        for i in range(stopien1):
            wspolczynniki[i] = self.wspolczynniki[i] + other.wspolczynniki[i]

        for i in range(stopien1, stopien2):
            wspolczynniki[i] = other.wspolczynniki[i]

    return Wielomian(wspolczynniki)
```

Ponieważ ze względu na możliwość dodawnia wielomianów o różnych stopniach nie możemy dodać współczynników znajdujących się na tych samych indeksach ze sobą, bo otrzymamy w końcu błąd związany z wyjściem poza tablicę.
Aby tego uniknąć program wybiera wielomian większego stopnia i jego współczynniki łączy ze współczynnikami na odpowiadających potęgach drugiego wielomianu. Następnie kopiowane są pozostałe współczynniki większego wielomianu.
Metoda zwraca nową instancję wielomianu.

Odejmowanie wielomianów

```
def __sub__(self, other):
    """
    Funkcja odejmuje do siebie dwa wielomiany
```

```
    """
    wspolczynniki = None
    stopien1 = self.__stopien(self.wspolczynniki)
    stopien2 = self.__stopien(other.wspolczynniki)
```

Żeby w łatwy sposób odejmować wielomiany przeciążany został operator `-`. Zanim program przystępuje do odejmowania pobiera stopnie obu wielomianów.

```
    if stopien1 > stopien2:
        wspolczynniki = [None] * stopien1
        for i in range(stopien2):
            wspolczynniki[i] = self.wspolczynniki[i] - other.wspolczynniki[i]

        for i in range(stopien2, stopien1):
            wspolczynniki[i] = self.wspolczynniki[i]

    else:
        wspolczynniki = [None] * stopien2
        for i in range(stopien1):
            wspolczynniki[i] = self.wspolczynniki[i] - other.wspolczynniki[i]

        for i in range(stopien1, stopien2):
            wspolczynniki[i] = -other.wspolczynniki[i]

    return Wielomian(wspolczynniki)
```

W przeciwieństwie do dodawania, kolejność od którego odejmujemy który wielomian ma już znaczenie. W programie zawsze odejmowany jest wielomian oznaczony jako `other` od wielomianu `self`.
Podobnie jak przy dodawaniu stopień wielomianu ma znaczenie. Jeśli od wielomianu o większym stopniu odejmujemy o niższym, na końcu operacji musimy przepisać pozostałe współczynniki większego stopniem. Natomiast gdy wykonujemy operację odwrotną to musimy przepisać pozostałe współczynniki większego stopniem wielomianu jednocześnie zmieniając ich znak.
Metoda zwraca nową instancję wielomianu.

Mnożenie wielomianów

```
def __mul__(self, other):
    """
    Funkcja mnożąca ze sobą dwa wielomiany
    """
    stopien1 = self.__stopien(self.wspolczynniki)
    stopien2 = self.__stopien(other.wspolczynniki)
    wspolczynniki = [None] * (stopien1 + stopien2 - 1)
```

Żeby w łatwy sposób mnożyć wielomiany przeciążany został operator *. Zanim program przystępuje do mnożenia pobiera stopnie obu wielomianów. I tworzy tablicę na nowe współczynniki.
Gdy jeden wielomian ma A stopień, drubi B to w wyniku pomnożenia otrzymujemy wielomian stopnia A + B. Stopnie wielomianu nie zawierają w sobie wyrrazu wolnego więc dodajemy jedno miejsce. Natomiast w programie wielomian jest reprezentowany współczynnikami z wyrazem wolnym włącznie. Więc jesli mamy wielomiany o M i N współczynnikach to wynikiem mnożenia będzie wielomian o M + N - 1 współczynnikach.

```
for i in range(stopien1):
    for j in range(stopien2):
        if wspolczynniki[i+j] == None:
            wspolczynniki[i+j] = self.wspolczynniki[i] * other.wspolczynniki[j]
        else:
            wspolczynniki[i+j] += self.wspolczynniki[i] * other.wspolczynniki[j]

return Wielomian(wspolczynniki)
```

Mnożenie wielomianów polega na wymnożeniu każdego współczynnika jednego wielomianu z każdym drugiego. Gdy przyjąć liczenie stopnia wielomianu to współczynnik stojący przy i stopniu pomnożony ze współczynnikiem przy j stopniu da nam współczynnik przy i + j stopniu.

Obliczanie wartości wielomianu algorytmem Hornera

```
def horner(self, x):
    """
        Funkcja obliczająca wartość wielomianu algorytmem Hornera dla danego
argumentu
    """
    wynik = 0

    for i in range(self.__stopien(self.wspolczynniki)-1, -1, -1):
        wynik = wynik * x + self.wspolczynniki[i]

    return wynik
```

Funkcja w sposób iteracyjny oblicza wartość wielomianu.

Dodatkowe funkcje

```
def stopien(self):
    """
    Funkcja zwracająca stopień wielomianu
    """
    return self.__stopien(self.wspolczynniki) - 1
```

Przeciążony został także operator __str__ pozwalający wypisywać wielomian.

Złożoności

Złożoność pamięciowa

Do przechowania n-stopniowego wielomianu potrzebujemu n+1 komórek tablicy. Liczba ta jest stała więc złożoność pamięciowa jest równa: Ω(n) = O(n)

Złożoność czasowa

1. Operacja dodawania i odejmowania
   Przyjmując za operację domunującą odczyt z tablicy otrzymujemy m + n odczytań (wielomiany mogą różnić się długością, gdy takie same mamy 2n odczytań). Otrzymujemy więc złożoność O(m+n), a zakładając dodatkowo że m<=n otrzymujemy złożoność O(n)
2. Operacja mnożenia
   Przyjmując za operację dominującą odczyt z tablicy otrzymujemy n oczytań pętli wewnętrznej i m odczytań pętli wewnętrznej co daje nam O(n*m), a zakładając m<=n otrzymuejmy złożoność O(n^2)
3. Obliczanie wartości wielomianu
   Przyjmując za operację dominującą operację mnożenia, w algorytmie Hornera dokonujemy tylko n mnożeń co daje nam notację O(n). Gdybyśmy chcieli obliczać wartość wielomianu w tradycyjny sposób (wymnażanie do uzyskania odpowiedniej potęgi) utrzymalibyśmy złożoność O(n^2)

---

4. Sposób uruchomienia

Aby uruchomić program należy wywołać komendę:
`python wielomian.py`

Aby utworzyć wielomian należy utworzyć nowy obiekt klasy i jako parametr podać tablicę współczynników np.:

```
>>> w1 = Wielomian([2, 4, 0, 1])
>>> print(w1)
2 + 4x + x^3
```

Aby dodać, odjąć lub wymnożyć wielomiany używamy odpowiednio operatorów `+ - *` np.:

```
>>> w1 = Wielomian([2, 4, 0, 1])
>>> w2 = Wielomian([2, 4])
>>> w3 = w1 + w2
>>> print(w3)
```

```
4 + 8x + x^3
>>> w3 = w1 - w2
>>> print(w3)
x^3
>>> w4 = w3 * w2
>>> print(w4)
2x^3 + 4x^4
```

Aby obliczyć wartość wielomianu dla zadanego argumentu używamy metody horner(x), gdzie x jest argument dla któego chcemy wartość np.:

```
>>> w1 = Wielomian([2, 4, 0, 1])
>>> wynik = w1.horner(1)
>>> print(wynik)
7.0
```

Aby uzyskać stopień wielomianu używamy metody stopien() np.:

```
>>> w1 = Wielomian([2, 4, 0, 1])
>>> print(print(w1.stopien()))
3
```

---

5. Literatura

https://pl.wikipedia.org/wiki/Schemat_Hornera

---

6. Wymagania

Python - testowane na wersji **3.9.7**