Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği
Bölümü
Veri Yapıları

2020-2021 Güz Dönemi

İki Polinom Arasında Liste Veri Yapısı Kullanarak Toplama Çıkarma İşlemi Raporu

Öğretim	1. Öğretim ☐	2. Öğretim ☒
Numara	Ad ve Soyad	
358623	Fırat Kaan BİTMEZ	
330199	Berk KUYUMCU	
314042	Osman Selçuk KARSLI	
Ders Sorumlusu:	Öğr. Gör. Dr. Zafer YAVUZ	

İki Polinom Arasında Liste Veri Yapısı Kullanarak Toplama Çıkarma İşlemi

Problem Tanımı

İki polinom arasında liste veri yapısı kullanarak toplama çıkarma işlemi yapıyoruz.

Kodlama

```
p1 = [(4, 3), (8, 1), (3, 0)] # P1 Polinomu 3x^0 + 8x^1 + 4x^3
p2 = [(4, 3),(2, 1), (8, 7), (6, 0), (8, 9), (2, 31)] # P2 Polinomu 6x^0 + 2x^1 + 4x^3 + 8x^7 + 8x^9 + 2x^31


# Fark Fonksiyonu
def diffPoly(PolyOne, PolyTwo):
    result = []  # sonuc Listesi
    tempPolyOne = [] # P1 Polinomunun gecici listesi
    tempPolyTwo = [] # P2 Polinomunun gecici listesi
```

Eger tempPolyOne listesi ilk defa doldurulacaksa parametre olarak polyOne icerigini bu listeye tek tek atiyoruz.

```
    if len(tempPolyOne) == 0:
        for i in range(len(PolyOne)):
            tempPolyOne.append(PolyOne[i])
```

Eger tempPolyTwo listesi ilk defa doldurulacaksa parametre olarak polyTwo icerigini bu listeye tek tek atiyoruz.

```
    if len(tempPolyTwo) == 0:
        for i in range(len(PolyTwo)):
            tempPolyTwo.append(PolyTwo[i])

    '''
```

Elimizde iki tane liste var ve bizim bu listeleri gezmemiz gerekiyor, bunun icin de PolyOne ve PolyTwo'nun uzunlugu kadar listeleri geziyoruz. Listeleri (carpan, us)olarak düzenlersek usleri eşit ise carpanlari cikararak bunu result

listesine ekliyoruz ve cikardigimiz iki değeri hem tempPolyOne hem de tempPolyTwo'dan siliyoruz.

```
for i in range(len(PolyOne)):
    for j in range(len(PolyTwo)):
        if PolyOne[i][1] == PolyTwo[j][1]:
            equalResult = PolyOne[i][0] - PolyTwo[j][0]
            result.append((equalResult, PolyOne[i][1]))
            tempPolyOne.remove((PolyOne[i][0], PolyOne[i][1]))
            tempPolyTwo.remove((PolyTwo[j][0], PolyTwo[j][1]))

'''
```

Eger tempPolyOne gecici listede usleri esit olmayan deger varsa bu degeri gidip result listesine ekliyoruz.

```
if len(tempPolyOne) != 0:
    for i in range(len(tempPolyOne)):
        result.append(tempPolyOne[i])
```

Eger tempPolyTwo gecici listede usleri esit olmayan deger varsa bu degeri gidip result listesine ekliyoruz.

```
if len(tempPolyTwo) != 0:
    for i in range(len(tempPolyTwo)):
        result.append(tempPolyTwo[i])
'''
    sonucu dondururken de result listesini usse gore siralayarak geriye donduruyoruz.
'''
return sorted(result, key=lambda x: x[1],reverse=True)
```

Burda polinomları üslerine göre azalan sırada sıralamak için reverse=True kullandık.

```
# Toplama Fonksiyonu
def addPoly(PolyOne, PolyTwo):
    result = []  # sonuc Listesi
    tempPolyOne = [] # P1 Polinomunun gecici listesi
    tempPolyTwo = [] # P2 Polinomunun gecici listesi
```

Eger tempPolyOne lsitesi ilk defa doldurulacaksa parametre olarak polyOne icerigini bu listeye tek tek atiyoruz.

```
if len(tempPolyOne) == 0:
    for i in range(len(PolyOne)):
        tempPolyOne.append(PolyOne[i])

...
```

Eger tempPolyTwo lsitesi ilk defa doldurulacaksa parametre olarak polyTwo icerigini bu listeye tek tek atiyoruz.

```
if len(tempPolyTwo) == 0:
    for i in range(len(PolyTwo)):
        tempPolyTwo.append(PolyTwo[i])
```

Elimizde iki tane liste var ve bizim bu listeleri gezmemiz gerekiyor bunun icin de PolyOne ve PolyTwo uzunlugu kadar listeleri geziyoruz. Listeleri (carpan, us) olarak duzenlersek; usleri esitise carpanlari toplayarak bunu result listesine ekliyoruz ve topladigimiz iki degeri hem tempPolyOne hem de tempPolyTwo'dan siliyoruz.

```
for i in range(len(PolyOne)):
    for j in range(len(PolyTwo)):
        if PolyOne[i][1] == PolyTwo[j][1]:
            equalResult = PolyOne[i][0] + PolyTwo[j][0]
            tempPolyOne.remove((PolyOne[i][0], PolyOne[i][1]))
            tempPolyTwo.remove((PolyTwo[j][0], PolyTwo[j][1]))
            result.append((equalResult, PolyOne[i][1]))
```

Eger tempPolyOne gecici listede usleri esit olmayan deger varsa Bu degeri gidip result listesine ekliyoruz.

```
if len(tempPolyOne) != 0:
    for i in range(len(tempPolyOne)):
        result.append(tempPolyOne[i])
```

Eger tempPolyTwo gecici listede usleri esit olmayan deger varsa Bu degeri gidip result listesine ekliyoruz.

```
if len(tempPolyTwo) != 0:
    for i in range(len(tempPolyTwo)):
        result.append(tempPolyTwo[i])
```

sonucu dondururken de result listesini usse gore siralayarak geriye donduruyoruz.

```
    return sorted(result, key=lambda x: x[1],reverse=True)

print(diffPoly(p1, p2))
print(addPoly(p1, p2))
```

1. ZAMAN KARMAŞIKLIĞI (TIME COMPLEXITY)

```
for i in range(len(PolyOne)):
    for j in range(len(PolyTwo)):
        if PolyOne[i][1] == PolyTwo[j][1]:
            equalResult = PolyOne[i][0] + PolyTwo[j][0]
            tempPolyOne.remove((PolyOne[i][0], PolyOne[i][1]))
            tempPolyTwo.remove((PolyTwo[j][0], PolyTwo[j][1]))
            result.append((equalResult, PolyOne[i][1]))
```

İçerdeki for n+1 dışardaki for n+1 olarak aldığımızda zamansal karmaşıklığımız (n+1).(n+1) ‘ den ;

T(n) = n^2 +2n+1 bulduk.

2. BIG O NOTASYONU

Sonuç olarak her zaman için algoritmamızın olabildiğince az zaman harcamasını isteriz. Fakat, bazen bu durum çok da mümkün olmayabilir, yine de yapabildiğimiz kadar girdiden bağımsız hale getirebilirsek, algoritmamız zaman açısından daha optimize hale gelecektir.

Büyük O Notasyonunun hesaplanması genel olarak şu şekildedir:
O(N + 1) -> O(N) olarak alınır.
O(N) + 1 -> O(N) olarak alınır.
O(N) * O(N) -> O(N^2) olarak alınır.

İç içe 2 for döngüsü olduğu için BIG O notasyonumuzu O(N^2) olarak bulduk.

Github Linki : https://github.com/Veri-Yapilari

Youtube Linki : https://youtu.be/BKkTC2Hxr1A

KAYNAKLAR

https://www.yazilimbilimi.org/python-liste-veri-tipi/

https://stackoverflow.com/questions/39093916/add-and-multiplication-of-polynomials-in-python

İŞ PAYLAŞIMI

Ekip olarak discord üzerinden iletişime geçtik.Ekran paylaşımı yaparak kodun algoritmasını birlikte düşündük ve ortak kararlarla kodu son haline getirdik.

Daha sonra karmaşıklığı ve notasyonu belirli kaynaklardan araştırarak çözüme ulaştırmaya çalıştık.

Discord üzerinden ödev hakkında açıklamamızı yaparak videoyu çektik.