Maiatza

May 28, 2021

1 2021-ko Maiatzeko Ohiko Deialdiko Azterketa

1.1 1. Ariketa (3 puntu)

Sortu itzazu ondoko funtzioak:

- **1a** - Zenbaki arrunt bat adierazten duen s karaktere kate bat jaso eta 2 digito ezberdin edo gehiago ote dituen bueltatuko duen funtzioa. Adibidez `'27722'` kateak soilik 2 digito ezberdin ditu eta `'4019'` kateak ordea 4

[1]:
```
def f1a(s):
    return len(set(s)) >= 2
```

[2]:
```
# Exekuzio adibidea (HAU EZ DA ESKATZEN)
for x in ('27722','4019','27', '277', '22', '22222222') :
    print(f'f1a({repr(x)}): {f1a(x)}')
```

```
f1a('27722'): True
f1a('4019'): True
f1a('27'): True
f1a('277'): True
f1a('22'): False
f1a('22222222'): False
```

- **1b** - Hutsunez banandutako zenbaki arruntez osotutako testu fitxategi baten bideizena eta bere kodifikazioa jaso, eta hiztegi bat bueltatuko duen funtzioa. Hiztegiaren gakoak 2 digito ezberdin edo gehiago dituzten **zenbaki osoak** izango dira eta balioak zenbaki bakoitzaren agerpen kopurua.

[3]:
```
def f1b(filename,encoding='utf8'):
    with open(filename, encoding=encoding) as f:
        h = {}
        for line in f :
            for w in line.split():
                if f1a(w):
                    i = int(w)
                    h[i] = h.get(i,0) + 1
        return h
```

[4]:
```
# Exekuzio adibidea (HAU EZ DA ESKATZEN)
f1b('ZenbakiArruntak.txt')
```

[4]:
```
{12: 4, 123: 5, 23: 3, 1224: 1, 213: 1, 112: 2, 1231: 2, 2312: 1, 32: 1}
```

- **1c** - Aurreko funtzioak bueltatzen duen moduko hiztegi bat jaso eta hiztegi berri bat bueltatuko duen funtzioa. Hiztegiaren gakoak zenbakien digito ezberdin kopuruak izango dira eta balioak, digito ezberdin kopuru hori duten **zenbaki oso zerrenda**.

[5]:
```
def f1c(h):
    h2 = {}
    for i in h :
        ndig = len(set(str(i)))
        h2.setdefault(ndig,[]).append(i)
    return h2
```

[6]:
```
# Exekuzio adibidea (HAU EZ DA ESKATZEN)
f1c(f1b('ZenbakiArruntak.txt'))
```

[6]:
```
{2: [12, 23, 112, 32], 3: [123, 1224, 213, 1231, 2312]}
```

1.2 2. Ariketa (2 puntu)

Demagun ondoko funtzioa dugula, zenbaki errealez osotutako z zerrenda jasotzen duena eta bi edozein elementuen arteko distantzia maximoa bueltatzen duena:

```
def max_dist(z):
    dmax = abs(z[0]-z[1])
    for i in range(len(z)-1):
        for j in range(i+1, len(z)):
            d = abs(z[i]-z[j])
            if d > dmax:
                dmax = d
    return dmax
```

- **2a** - Aztertu funtzioaren konplexutasun tenporala, zerrendaren `n=len(z)` tamainarekiko, notazio asintotikoan adieraziz.

Funtzioak ez du kasu on edo txarrik, beraz orden zehatza kalkulatuko dugu zuzenean:

```
def max_dist(z):
    dmax = abs(z[0]-z[1])              # 1 pausu
    for i in range(len(z)-1):          # (n-1) x 1 pausu
        for j in range(i+1, len(z)):   # (n-1) + (n-2) + ... + 2 + 1 pausu
            d = abs(z[i]-z[j])         # (n-1) + (n-2) + ... + 2 + 1 pausu
            if d > dmax:               # (n-1) + (n-2) + ... + 2 + 1 pausu
                dmax = d               # aurrekoaren barnean
    return dmax                        # 1 pausu
```

Beraz, denera:

$$t(n) = 2 + (n-1) + 3 \cdot (n-1+n-2+\cdots+2+1) = 1 + n + \frac{3n \cdot (n-1)}{2} \quad \in \Theta(n^2)$$

Funtzioak `for i in range(...)` moduko kontrol egiturak erabiltzen dituenes, $t(n)$ pausu kopurua zuzenean batukarien bidez ere lor daiteke:

$$t(n) = 1 + \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = 1 + \sum_{i=0}^{n-2} (n-i-1) \underset{k=n-i-1}{=} 1 + \sum_{k=n-1}^{1} k = 1 + \frac{(n-1) \cdot n}{2} \quad \in \Theta(n^2)$$

- **2b** - Saia zaitez algoritmo bizkorrago bat sortzen eta bere konplexutasun tenporala aztertu ezazu.

Bi algoritmo bizkorrago jarriko ditugu, lehenengoa $\Theta(n \log n)$ eta bigarrena $\Theta(n)$. Argi eta garbi, bigarrena da zuzenena.

[7]:
```
def max_dist(z):
    # kontuz z.sort() egitearekin
    z = sorted(z)       # (n + n x log n) pausu: n kopia egiteko eta nxlogn␣
↪ordenatzeko
    return z[-1]-z[0]   # 1 pausu
```

[8]:
```
def max_dist(z):
    return max(z) - min(z)  # (2 x n + 1) pausu: n max eta min kalkulatzeko eta␣
↪1 kenketarako
```

1.3 3. Ariketa (puntu 1)

`z` zerrenda batetatik `x` balio baten agerpen guztiak ezabatuko dituen funtzioa sortu ezazu. Funtzioak $\Theta(n)$ orden zehatza izan beharko du, `n=len(z)` zerrendaren tamaina delarik. Funtzioak ez du zerrenda berri bat bueltatuko, jasotako zerrendan aldaketak egingo dituelarik.

[9]:
```
def f3(z,x):
    aux = [y for y in z if y != x]
    z.clear()
    z.extend(aux)
```

[10]:
```
# Exekuzio adibidea (HAU EZ DA ESKATZEN)
z = [1,2,3,4,5,4,3,2,1,2,3,2,1,2,3,1]
f3(z,3)
z
```

[10]:
```
[1, 2, 4, 5, 4, 2, 1, 2, 2, 1, 2, 1]
```

1.4 4. Ariketa (4 puntu)

Sortu ezazu $\frac{a}{b}$ moduko zatikiak (a eta b zenbaki osoak izanik) adieraziko dituen `Fraction` klasea. Klaseak ondoko metodoak izango ditu:

- `__init__(self,a,b)` $\frac{a}{b}$ zatiki berri bat sortuko du eta ondorengo baldintzak beteko dira:
- $b = 0$ balitz `ValueError` motako salbuespena gertatuko da.
- Zatikiaren zeinua zatikizunak eramango du. Hau da, b zatitzailea beti balio positiboarekin geldituko da.
- Zatikia forma laburtuan geldituko da. Hau da, $a \neq 0$ izanik, $i = zkh(|a|, |b|)$ $|a|$ eta $|b|$-ren Zatitzaile Komunetako Handiena bada, orduan gordeko den forma laburtua $\frac{a/i}{b/i}$ izango da. $a = 0$ bada, *zatiki nulua* dela diogu eta zatikia $\frac{0}{1}$ moduan geldituko da.
- `__add__(self, other)` Bi zatikien batura (zatiki bat) bueltatzen du.
- `__sub__(self, other)` Bi zatikien kenketa (zatiki bat) bueltatzen du.
- `__mul__(self, other)` Bi zatikien biderkadura (zatiki bat) bueltatzen du.
- `__truediv__(self, other)` Bi zatikien zatidura (zatiki bat) bueltatzen du. `other` zatikia $\frac{0}{1}$ *zatiki nulua* bada, `ZeroDivisionError` motako salbuespena gertatuko da.
- `__invert__(self)` Alderantzizko zatikia bueltatzen du. `self` zatikia $\frac{0}{1}$ *zatiki nulua* bada, `ZeroDivisionError` motako salbuespena gertatuko da.
- `__neg__(self)` Alderantzizko zeinua duen zatikia bueltatzen du.
- `__eq__(self, other)` Bi zatiki berdinak ote diren bueltatzen du.
- `__str__(self)` Zatikiaren testu errepresentazio bat bueltatzen du.
- `__repr__(self)` Zatikiaren testu errepresentazio kanonikoa bueltatzen du.

[11]:
```python
class Fraction(object):

    def __init__(self,a,b):
        if b == 0 :
            raise ValueError('Zero divisor')
        if b < 0:
            a,b = -a,-b
        if a != 0 :
            i = Fraction.gcd(a,b)
            a,b = a//i,b//i
        else :
            b = 1
        self.a,self.b = a,b

    def gcd(i,j):
        r = i%j
        return j if r==0 else Fraction.gcd(j,r)
```

```
    def __add__(self,other):
        return Fraction(self.a*other.b+self.b*other.a, self.b*other.b)

    def __sub__(self,other):
        return Fraction(self.a*other.b-self.b*other.a, self.b*other.b)

    def __mul__(self,other):
        return Fraction(self.a*other.a, self.b*other.b)

    def __bool__(self):
        return bool(self.a)

    def __truediv__(self,other):
        if other :
            return Fraction(self.a*other.b, self.b*other.a)
        else:
            raise ZeroDivisionError("Division by null fraction")

    def __invert__(self):
        if self :
            return Fraction(self.b, self.a)
        else:
            raise ZeroDivisionError("Could not invert null fraction")

    def __neg__(self):
        return Fraction(-self.a,self.b)

    def __eq__(self,other):
        return type(other)==Fraction and self.a == other.a and self.b == other.b

    def __str__(self):
        return f'{self.a}/{self.b}'

    def __repr__(self):
        return f'Fraction({self.a},{self.b})'
```

[12]:
```
# Exekuzio adibidea (HAU EZ DA ESKATZEN)

z = [(1,2),(-5,-10),(10,-20),(10,25),(24,42),(34234,1312123512),(0,34)]
F = [Fraction(a,b) for a,b in z]

# str frogatu
print("-"*20,'str',"-"*20)
for (a,b),f in zip(z,F):
    print(f'Fraction({a},{b}) = {f}')

# repr<-->eval frogatu
```

```
print("-"*20,'repr/eval',"-"*20)
print(all(eval(repr(f)) == f for f in F))

# batura frogatu
print("-"*20,'add',"-"*20)
for f1,f2 in zip(F,F[1:]):
    print(f'{f1} + {f2} = {f1+f2}')

# kenketa frogatu
print("-"*20,'sub',"-"*20)
for f1,f2 in zip(F,F[1:]):
    print(f'{f1} - {f2} = {f1-f2}')

# biderkadura frogatu
print("-"*20,'mul',"-"*20)
for f1,f2 in zip(F,F[1:]):
    print(f'{f1} * {f2} = {f1*f2}')

# zatidura frogatu
print("-"*20,'truediv',"-"*20)
for f1,f2 in zip(F,F[1:]):
    try:
        print(f'{f1} / {f2} = {f1/f2}')
    except ZeroDivisionError as e:
        print(f'{f1} / {f2} --> {e}')

# inbertsioa frogatu
print("-"*20,'invert',"-"*20)
for f1 in F:
    try:
        print(f'~ {f1} = {~f1}')
    except ZeroDivisionError as e:
        print(f'~ {f1} --> {e}')

# negazioa frogatu
print("-"*20,'neg',"-"*20)
for f1 in F:
    print(f'- {f1} = {-f1}')
```

```
-------------------- str --------------------
Fraction(1,2) = 1/2
Fraction(-5,-10) = 1/2
Fraction(10,-20) = -1/2
Fraction(10,25) = 2/5
Fraction(24,42) = 4/7
Fraction(34234,1312123512) = 17117/656061756
Fraction(0,34) = 0/1
```

```
-------------------- repr/eval --------------------
True
-------------------- add --------------------
1/2 + 1/2 = 1/1
1/2 + -1/2 = 0/1
-1/2 + 2/5 = -1/10
2/5 + 4/7 = 34/35
4/7 + 17117/656061756 = 53558507/93723108
17117/656061756 + 0/1 = 17117/656061756
-------------------- sub --------------------
1/2 - 1/2 = 0/1
1/2 - -1/2 = 1/1
-1/2 - 2/5 = -9/10
2/5 - 4/7 = -6/35
4/7 - 17117/656061756 = 374875315/656061756
17117/656061756 - 0/1 = 17117/656061756
-------------------- mul --------------------
1/2 * 1/2 = 1/4
1/2 * -1/2 = -1/4
-1/2 * 2/5 = -1/5
2/5 * 4/7 = 8/35
4/7 * 17117/656061756 = 17117/1148108073
17117/656061756 * 0/1 = 0/1
-------------------- truediv --------------------
1/2 / 1/2 = 1/1
1/2 / -1/2 = -1/1
-1/2 / 2/5 = -5/4
2/5 / 4/7 = 7/10
4/7 / 17117/656061756 = 374892432/17117
17117/656061756 / 0/1 --> Division by null fraction
-------------------- invert --------------------
~ 1/2 = 2/1
~ 1/2 = 2/1
~ -1/2 = -2/1
~ 2/5 = 5/2
~ 4/7 = 7/4
~ 17117/656061756 = 656061756/17117
~ 0/1 --> Could not invert null fraction
-------------------- neg --------------------
- 1/2 = -1/2
- 1/2 = -1/2
- -1/2 = 1/2
- 2/5 = -2/5
- 4/7 = -4/7
- 17117/656061756 = -17117/656061756
- 0/1 = 0/1
```