Универзитет у Београду

Електротехнички факултет

Одабрана поглавља нумеричке анализе

Пројектни задатак 1

Најбоље рационалне апроксимације реалних бројева

Студент:

Александра Богићевић 0390/17

Београд, школска година 2021/2022

Задатак:

Нека је дат позитиван реалан број α са коначним децималским записом и нека су дати природни бројеви n и m, тако да $n < m$. Формирати низ разломака p/q таквих да за именилац q важи $n \leq q \leq m$ (тј. $q = n, n + 1, . . . ,m$) и при том имениоцу q придружујемо бројилац p који одређујемо заокруживањем на најближи природан број производа $\alpha \cdot q$. Представити сваки разломак p/q у облику верижног разломака. У низу разломака p/q издвојити:

- најбоље рационалне апроксимације I врсте,
- најбоље рационалне апроксимације II врсте,
- сортирати разломке p/q по услову минималности апсолутне грешке $|\alpha - p/q|$.

Напомена:

За рачунање решења коришћен је програмски језик *Python* и његове библиотеке *numpy*,*math* и *decimal*. Радно окружење је *Jupyter notebook*. На дну документа се налази прилог са целокупним кодом.

```
import numpy as np
import math
from decimal import *
```

Решење:

За решавање овог проблема, пре свега потребно је формирати класу којом се представља разломак и све њене методе.

Конструктору се прослеђују вредности имениоца и бројиоца. Осим конструктора, класа садржи методе које израчунавају вредност апсолутне грешке, методу која представља разломак у верижном облику, и методе које олакшавају испис.

```
class Fraction: #p/q
    def __init__(self, p, q):
        self.p = p
        self.q = q

    def set_p(self, p): #brojilac
        self.p = p
    def set_q(self, q): #imenilac
        self.q = q

    def set_e1(self, alpha): #vrednost apsolutne greske, za racunanje prve vrste
        self.e1_sign = alpha-self.p/self.q
        self.e1 = np.absolute(alpha-self.p/self.q)

    def set_e2(self, alpha): #vrednost apsolutne greske, za racunanje druge vrste
        self.e2_sign = self.q*alpha-self.p
        self.e2 = np.absolute(self.q*alpha-self.p)

    def printFr(self):
        res = str(self.p) + "/" + str(self.q)
        return res
```

```python
def verizniRazlomak(self):  #algoritam za racunanje veriznog razlomka
    k = 10
    self.verizni = []
    x0 = Decimal(self.p/self.q)
    a0 = Decimal(math.floor(Decimal(x0)))
    d0 = Decimal(x0 - a0)
    x=[]
    x.append(x0)
    a=[]
    a.append(a0)
    self.verizni.append(a0)
    d=[]
    d.append(d0)


    for i in range(1,k):
        if d[i-1] <= 0.00000001:
            break
        xx = Decimal(1 / d[i-1])
        x.append(xx)

        aa = Decimal(math.floor(xx))
        a.append(aa)
        self.verizni.append(aa)

        dd = Decimal(xx - aa)
        d.append(dd)

    last_index = len(a)-1
    if a[last_index] == 1:
        a[last_index - 1] = a[last_index - 1] + 1
        self.verizni[last_index - 1] = self.verizni[last_index - 1] + 1
        del a[last_index]
        del self.verizni[last_index]

    return
```

```python
def printVerizni(self):
    res = "["
    for i in range(len(self.verizni)):
        res = res + str(self.verizni[i])
        if(i==0 and len(self.verizni)>1):
            res = res + ";"
        else:
            if i!=len(self.verizni)-1:
                res = res + ","
    res = res + "]"
    return res
```

Такође је потребно увести и низове за смештање свих разломака, најбоље рационалне апроксимације прве врсте и најбоље рационалне апроксимације друге врсте.

```
all_fractions = []
best_first = []
best_second = []
```

На почетку програма тражимо унос података (*α*, *m*, *n*).

Пример 1:

Разматратрамо пример универзалне параболичке константе (***universal parabolic constant***) која износи **2.2955871**.

Наши улазни параметри су:

α = 2.2955871

n = 1

m = 15

```
badInput = True
while badInput:
    alpha = float(input("Uneti realan broj: "))
    n = int(input("Uneti vrednost n: "))
    m = int(input("Uneti vrednost m, tako da je n<m: "))
    if(n>=m):
        print("Nije ispunjen uslov n<m! Ponovite unos.")
    else:
        badInput = False
```

Излаз програма:

```
Uneti realan broj: 2.2955871
Uneti vrednost n: 1
Uneti vrednost m, tako da je n<m: 15
```

Када имамо унете вредности можемо да формирамо низ разломака *p*/*q* који је тражен у задатку.

```
for q in range(n, m+1):
    p = int(np.round(alpha*q))
    fr = Fraction(p, q)
    fr.set_e1(alpha)
    fr.set_e2(alpha)
    fr.veriznIRazlomak()
    all_fractions.append(fr)
```

При формирању низа разломака може се приметити да се одмах и позивају методе које рачунају апсолутну грешку, као и метода која разломак представља у верижном облику.

У наставку следи изглед формираног низа.

```
for i in range(len(all_fractions)):
    print(all_fractions[i].printFr() + " " + all_fractions[i].printVerizni()  +
          "  e1 = "+ str(all_fractions[i].e1_sign) +"  e2 = "+ str(all_fractions[i].e2_sign))
```

```
2/1 [2]  e1 = 0.2955871000000001  e2 = 0.2955871000000001
5/2 [2;2]  e1 = -0.2044128999999999  e2 = -0.4088257999999998
7/3 [2;3]  e1 = -0.03774623333333338  e2 = -0.11323870000000014
9/4 [2;4]  e1 = 0.0455871000000001  e2 = 0.1823484000000004
11/5 [2;5]  e1 = 0.09558709999999992  e2 = 0.47793550000000096
14/6 [2;3]  e1 = -0.03774623333333338  e2 = -0.22647740000000027
16/7 [2;3,2]  e1 = 0.009872814285714515  e2 = 0.06910970000000205
18/8 [2;4]  e1 = 0.0455871000000001  e2 = 0.3646968000000008
21/9 [2;3]  e1 = -0.03774623333333338  e2 = -0.3397161000000004
23/10 [2;3,3]  e1 = -0.00441289999999972  e2 = -0.044128999999998086
25/11 [2;3,1,2]  e1 = 0.022859827272727173  e2 = 0.2514581000000007
28/12 [2;3]  e1 = -0.03774623333333338  e2 = -0.45295480000000055
30/13 [2;3,4]  e1 = -0.012105207692307385  e2 = -0.15736769999999822
32/14 [2;3,2]  e1 = 0.009872814285714515  e2 = 0.1382194000000041
34/15 [2;3,1,3]  e1 = 0.028920433333333495  e2 = 0.43380650000000287
```

Да бисмо испратили ток решења, можемо представити и број α у верижном облику.

```
k = m-n
alpha_verizni=[]

x0 = Decimal(alpha)
a0 = Decimal(math.floor(Decimal(x0)))
d0 = Decimal(x0 - a0)
x=[]
x.append(x0)
a=[]
a.append(a0)
alpha_verizni.append(a0)
d=[]
d.append(d0)


for i in range(1,k):
    if d[i-1] <= 0.000000000000000000000000000000000000000001:
        break
    xx = Decimal( 1 / d[i-1])
    x.append(xx)
    aa = Decimal(math.floor(Decimal(xx)))
    a.append(aa)
    alpha_verizni.append(aa)

    dd = Decimal(xx - aa )
    d.append(dd)

last_index = len(a)-1
if a[last_index] == 1:
    a[last_index - 1] = a[last_index - 1] + 1
    alpha_verizni[last_index - 1] = alpha_verizni[last_index - 1] + 1
    del a[last_index]
    del alpha_verizni[last_index]

res = "["
for i in range(len(alpha_verizni)):
    res = res + str(alpha_verizni[i])
    if(i==0 and len(alpha_verizni)>1):
        res = res + ";"
    else:
        if i!=len(alpha_verizni)-1:
            res = res + ","
res = res + "]"
print(res)
```

Излаз програма:

```
[2;3,2,1,1,1,1,3,3,1,1,7,5]
```

Формиравши низ свих разломака, можемо из њега издвојити најбоље апроксимације прве и друге врсте.

Рационални број p/q је **најбоља апроксимација прве врсте** броја α ако важи

$$\left|\alpha - \frac{p}{q}\right| < \left|\alpha - \frac{r}{s}\right|$$

за све разломке $\frac{r}{s} \neq \frac{p}{q}$ такве да је $0 < s \leq q$.

Рационални број p/q је **најбоља апроксимација друге врсте** броја α ако важи

$$|q\alpha - p| < |s\alpha - r|$$

за све разломке $\frac{r}{s} \neq \frac{p}{q}$ такве да је $0 < s \leq q$.

Уводимо променљиве у којима ћемо памтити досадашњи минимум грешке. Променљиве иницијализујемо на највећу могућу вредност.

```
e1_min = float("inf")
e2_min = float("inf")
```

На крају покрећемо алгоритам за разврставање разломака:

```
for i in range(0, len(all_fractions)):
    fr = all_fractions[i]
    if fr.e2 < e2_min:
        e2_min = fr.e2
        best_second.append(fr)
        if fr.e1 < e1_min:
            e1_min = fr.e1
        continue
    if fr.e1 < e1_min:
        e1_min = fr.e1
        best_first.append(fr)
```

Пошто су најбоље рационалне апроксимације друге врсте уједно и најбоље апроксимације прве врсте, прво се врши провера да ли разломак припада другој врсти, и ако припада њој се приписује. Потом се врши провера да ли разломак припада првој врсти.

На крају овог алгоритма имамо формирана сва 3 низа.

Да бисмо испратили извршавање програма, исписаћемо садржај два новонастала низа.

Апроксимације прве врсте:

```
print("APROKSIMACIJE PRVE VRSTE:")
for fr in best_first:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))

print("Sortirano:")
sorted_best_first = sorted(best_first, key=lambda fr : fr.e1)
for fr in sorted_best_first:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))
```

Излаз програма:

```
APROKSIMACIJE PRVE VRSTE:
5/2 [2;2]  e1 = -0.2044128999999999  e2 = -0.4088257999999998
Sortirano:
5/2 [2;2]  e1 = -0.2044128999999999  e2 = -0.4088257999999998
```

Апроксимације друге врсте:

```
print("APROKSIMACIJE DRUGE VRSTE:")
for fr in best_second:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))

print("Sortirano:")
sorted_best_second = sorted(best_second, key=lambda fr : fr.e1)
for fr in sorted_best_second:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))
```

Излаз програма:

```
APROKSIMACIJE DRUGE VRSTE:
2/1 [2]  e1 = 0.2955871000000001  e2 = 0.2955871000000001
7/3 [2;3]  e1 = -0.03774623333333338  e2 = -0.11323870000000014
16/7 [2;3,2]  e1 = 0.009872814285714515  e2 = 0.06910970000000205
23/10 [2;3,3]  e1 = -0.00441289999999972  e2 = -0.04412899999998086
Sortirano:
23/10 [2;3,3]  e1 = -0.00441289999999972  e2 = -0.04412899999998086
16/7 [2;3,2]  e1 = 0.009872814285714515  e2 = 0.06910970000000205
7/3 [2;3]  e1 = -0.03774623333333338  e2 = -0.11323870000000014
2/1 [2]  e1 = 0.2955871000000001  e2 = 0.2955871000000001
```

Како бисмо одговорили на задатак потребно је сортирати све разломке по услову минималности апсолутне грешке $|\alpha - p/q|$.

```
sorted_all_fr = sorted(all_fractions, key=lambda fr : fr.e1)
```

Сада можемо да испишемо коначне резултате. При испису, последња колона означава којој врсти апроксимације разломак припада, при чему *I* означава најбоље апроксимације прве врсте, *II* означава најбоље апроксимације друге врсте, а *N* означава апроксимације које не припадају ни првој ни другој врсти.

```
for fr in sorted_all_fr:
    if fr in best_first:
        print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) + "   I")
    elif fr in best_second:
         print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) + "  II")
    else:
         print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+ str(fr.e1_sign) + "  N")
```

Излаз програма:

```
23/10 [2;3,3]  e1 = -0.00441289999999972  II
16/7 [2;3,2]  e1 = 0.009872814285714515  II
32/14 [2;3,2]  e1 = 0.009872814285714515  N
30/13 [2;3,4]  e1 = -0.012105207692307385  N
25/11 [2;3,1,2]  e1 = 0.022859827272727173  N
34/15 [2;3,1,3]  e1 = 0.028920433333333495  N
7/3 [2;3]  e1 = -0.037746233333333338  II
14/6 [2;3]  e1 = -0.037746233333333338  N
21/9 [2;3]  e1 = -0.037746233333333338  N
28/12 [2;3]  e1 = -0.037746233333333338  N
9/4 [2;4]  e1 = 0.0455871000000001  N
18/8 [2;4]  e1 = 0.0455871000000001  N
11/5 [2;5]  e1 = 0.09558709999999992  N
5/2 [2;2]  e1 = -0.2044128999999999   I
2/1 [2]  e1 = 0.2955871000000001  II
```

Пример 2:

Наши улазни параметри су:

$\alpha = \log_2 3 - 1 = 0.5849625$

n = 5

m = 15

Испис свих разломака:

```
3/5 [0;1,1,2]  e1 = -0.015037499999999926  e2 = -0.07518749999999974
4/6 [0;1,2]  e1 = -0.08170416666666658  e2 = -0.4902249999999997
4/7 [0;1,1,3]  e1 = 0.013533928571428655  e2 = 0.09473750000000081
5/8 [0;1,1,1,2]  e1 = -0.040037499999999995  e2 = -0.3202999999999996
5/9 [0;1,1,4]  e1 = 0.029406944444444447  e2 = 0.2646625
6/10 [0;1,1,2]  e1 = -0.015037499999999926  e2 = -0.15037499999999948
6/11 [0;1,1,5]  e1 = 0.03950795454545464  e2 = 0.434587500000001
7/12 [0;1,1,2,2]  e1 = 0.0016291666666666815  e2 = 0.019550000000000622
8/13 [0;1,1,1,1,2]  e1 = -0.030422115384615367  e2 = -0.39548749999999977
8/14 [0;1,1,3]  e1 = 0.013533928571428655  e2 = 0.18947500000000161
9/15 [0;1,1,2]  e1 = -0.015037499999999926  e2 = -0.22556249999999878
```

Верижни облик константе:

```
[0;1,1,2,2,3,1,5,2,23]
```

Најбоље рационалне апроксимације прве врсте:

```
APROKSIMACIJE PRVE VRSTE:
4/7 [0;1,1,3]  e1 = 0.013533928571428655  e2 = 0.09473750000000081
Sortirano:
4/7 [0;1,1,3]  e1 = 0.013533928571428655  e2 = 0.09473750000000081
```

Најбоље рационалне апроксимације друге врсте:

```
APROKSIMACIJE DRUGE VRSTE:
3/5 [0;1,1,2]  e1 = -0.015037499999999926  e2 = -0.07518749999999974
7/12 [0;1,1,2,2]  e1 = 0.0016291666666666815  e2 = 0.019550000000000622
Sortirano:
7/12 [0;1,1,2,2]  e1 = 0.0016291666666666815  e2 = 0.019550000000000622
3/5 [0;1,1,2]  e1 = -0.015037499999999926  e2 = -0.07518749999999974
```

Коначан испис:

```
7/12 [0;1,1,2,2]  e1 = 0.0016291666666666815  II
4/7 [0;1,1,3]  e1 = 0.013533928571428655   I
8/14 [0;1,1,3]  e1 = 0.013533928571428655  N
3/5 [0;1,1,2]  e1 = -0.015037499999999926  II
6/10 [0;1,1,2]  e1 = -0.015037499999999926  N
9/15 [0;1,1,2]  e1 = -0.015037499999999926  N
5/9 [0;1,1,4]  e1 = 0.02940694444444447  N
8/13 [0;1,1,1,1,2]  e1 = -0.030422115384615367  N
6/11 [0;1,1,5]  e1 = 0.03950795454545464  N
5/8 [0;1,1,1,2]  e1 = -0.04003749999999995  N
4/6 [0;1,2]  e1 = -0.08170416666666658  N
```

Може се приметити да овако написан алгоритам не региструје умношке разломака, већ их означава са *N*. То је из простог разлога што је дефиниција таква да ће се за различите умношке разликовати апсолутна грешка при рачунању најбоље апроксимације друге врсте, и такође се користи знак строго мање („<“) при поређењу грешака.

То можемо превазићи уколико пре рачунања апсолутне грешке редукујемо разломак, тако што нађемо највећи заједнички делилац за *p* и *q*, и поделимо и *p* и *q* њиме, и када бисмо знак „<“ за поређење минималне грешке заменили знаком „≤“, међутим то у овом раду није рађено јер смо се држали дефиниције.

Пример 3:

Наши улазни параметри су:

α = π = 3.1415926

n = 1

m = 10

Испис свих разломака:

```
3/1 [3]  e1 = 0.14159260000000007  e2 = 0.14159260000000007
6/2 [3]  e1 = 0.14159260000000007  e2 = 0.28318520000000014
9/3 [3]  e1 = 0.14159260000000007  e2 = 0.4247778000000011
13/4 [3;4]  e1 = -0.10840739999999993  e2 = -0.4336295999999997
16/5 [3;5]  e1 = -0.05840740000000011  e2 = -0.29203700000000055
19/6 [3;6]  e1 = -0.02507406666666645  e2 = -0.1504443999999978
22/7 [3;7]  e1 = -0.0012645428571427253  e2 = -0.008851799999998633
25/8 [3;8]  e1 = 0.01659260000000007  e2 = 0.13274080000000055
28/9 [3;9]  e1 = 0.030481488888888908  e2 = 0.2743333999999997
31/10 [3;10]  e1 = 0.04159259999999998  e2 = 0.4159259999999989
```

Верижни облик константе:

```
[3;7,15,1,243,1,1,10]
```

Најбоље рационалне апроксимације прве врсте:

```
APROKSIMACIJE PRVE VRSTE:
13/4 [3;4]  e1 = -0.10840739999999993  e2 = -0.4336295999999997
16/5 [3;5]  e1 = -0.05840740000000011  e2 = -0.29203700000000055
19/6 [3;6]  e1 = -0.02507406666666645  e2 = -0.1504443999999978
Sortirano:
19/6 [3;6]  e1 = -0.02507406666666645  e2 = -0.1504443999999978
16/5 [3;5]  e1 = -0.05840740000000011  e2 = -0.29203700000000055
13/4 [3;4]  e1 = -0.10840739999999993  e2 = -0.4336295999999997
```

Најбоље рационалне апроксимације друге врсте:

```
APROKSIMACIJE DRUGE VRSTE:
3/1 [3]  e1 = 0.14159260000000007  e2 = 0.14159260000000007
22/7 [3;7]  e1 = -0.0012645428571427253  e2 = -0.008851799999998633
Sortirano:
22/7 [3;7]  e1 = -0.0012645428571427253  e2 = -0.008851799999998633
3/1 [3]  e1 = 0.14159260000000007  e2 = 0.14159260000000007
```

Коначан испис:

```
22/7 [3;7]  e1 = -0.0012645428571427253  II
25/8 [3;8]  e1 = 0.01659260000000007  N
19/6 [3;6]  e1 = -0.02507406666666645   I
28/9 [3;9]  e1 = 0.030481488888888908  N
31/10 [3;10]  e1 = 0.04159259999999998  N
16/5 [3;5]  e1 = -0.05840740000000011   I
13/4 [3;4]  e1 = -0.10840739999999993   I
3/1 [3]  e1 = 0.14159260000000007  II
6/2 [3]  e1 = 0.14159260000000007  N
9/3 [3]  e1 = 0.14159260000000007  N
```

Прилог:

У наставку следи приказ кода у целости.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import math
from decimal import *


# Uvodimo klasu kojom predstavljamo razlomak (razlomak je oblika p/q):

# In[2]:


class Fraction: #p/q
    def __init__(self, p, q):
        self.p = p
        self.q = q

    def set_p(self, p): #brojilac
        self.p = p
    def set_q(self, q): #imenilac
        self.q = q

    def set_e1(self, alpha): #vrednost apsolutne greske, za racunanje prve
vrste
        self.e1_sign = alpha-self.p/self.q
        self.e1 = np.absolute(alpha-self.p/self.q)

    def set_e2(self, alpha): #vrednost apsolutne greske, za racunanje druge
vrste
        self.e2_sign = self.q*alpha-self.p
        self.e2 = np.absolute(self.q*alpha-self.p)

    def printFr(self):
        res = str(self.p) + "/" + str(self.q)
        return res

    def verizniRazlomak(self):  #algoritam za racunanje veriznog razlomka
        k = 10
        self.verizni = []
        x0 = Decimal(self.p/self.q)
        a0 = Decimal(math.floor(Decimal(x0)))
        d0 = Decimal(x0 - a0)
        x=[]
        x.append(x0)
        a=[]
        a.append(a0)
        self.verizni.append(a0)
        d=[]
```

```
        d.append(d0)


        for i in range(1,k):
            if d[i-1] <= 0.00000001:
                break
            xx = Decimal(1 / d[i-1])
            x.append(xx)

            aa = Decimal(math.floor(xx))
            a.append(aa)
            self.verizni.append(aa)

            dd = Decimal(xx - aa)
            d.append(dd)

        last_index = len(a)-1
        if a[last_index] == 1:
            a[last_index - 1] = a[last_index - 1] + 1
            self.verizni[last_index - 1] = self.verizni[last_index - 1] + 1
            del a[last_index]
            del self.verizni[last_index]

        return

    def printVerizni(self):
        res = "["
        for i in range(len(self.verizni)):
            res = res + str(self.verizni[i])
            if(i==0 and len(self.verizni)>1):
                res = res + ";"
            else:
                if i!=len(self.verizni)-1:
                    res = res + ","
        res = res + "]"
        return res


# Uvodimo nizove gde cemo smestati:
# all_fractions - sve razlomke p/q
# best_first - najbolje racionalne aproksimacije prve vrste
# best_second - najbolje racionalne aproksimacije druge vrste

# In[3]:


all_fractions = []
best_first = []
best_second = []


# Uvodimo promenljivu u kojoj cemo cuvati minimalnu gresku i koji razlomak
daje tu gresku

# In[4]:
```

```
e1_min = float("inf")
e2_min = float("inf")


# Unosimo vrednosti alfa, n i m.
# alfa=2.2955871
# n = 1
# m = 15

# In[5]:


badInput = True
while badInput:
    alpha = float(input("Uneti realan broj: "))
    n = int(input("Uneti vrednost n: "))
    m = int(input("Uneti vrednost m, tako da je n<m: "))
    if(n>=m):
        print("Nije ispunjen uslov n<m! Ponovite unos.")
    else:
        badInput = False


# Algoritam za ispis razlomaka p/q:
# Formiramo niz razlomaka (all_fractions) oblika p/q, tako da je:
#     q = (n, n+1, n+2,..,m)
#     p = alpha * q, zaokruzeno na najblizi prirodan broj

# In[6]:


for q in range(n, m+1):
    p = int(np.round(alpha*q))
    fr = Fraction(p, q)
    fr.set_e1(alpha)
    fr.set_e2(alpha)
    fr.veriznIRazlomak()
    all_fractions.append(fr)


# Ispis razlomaka i greske:

# In[7]:


for i in range(len(all_fractions)):
    print(all_fractions[i].printFr() + " " + all_fractions[i].printVerizni()
+
          "  e1 = "+ str(all_fractions[i].e1_sign) +"  e2 = "+
str(all_fractions[i].e2_sign))



# Sortiranje i ispis liste po uslovu minimalnosti apsolutne greske |alpha-
p/q|:

# In[8]:
```

```
sorted_all_fr = sorted(all_fractions, key=lambda fr : fr.e1)
print("Sortirana lista razlomaka po minimalnosti apsolutne greske e=|alpha-
p/q| :")
for i in range(len(sorted_all_fr)):
    print(sorted_all_fr[i].printFr()+ " " + sorted_all_fr[i].printVerizni()
+"  e1 = "+ str(sorted_all_fr[i].e1_sign))


# Sortiranje i ispis liste po uslovu minimalnosti apsolutne greske |alpha*q-
p|:

# In[9]:


sorted_all_fr2 = sorted(all_fractions, key=lambda fr : fr.e2)
print("Sortirana lista razlomaka po minimalnosti apsolutne greske e=|alpha*q-
p| :")
for i in range(len(sorted_all_fr)):
    print(sorted_all_fr2[i].printFr()+ " " + sorted_all_fr2[i].printVerizni()
+"  e2 = "+ str(sorted_all_fr2[i].e2_sign))


# Trazimo verizne razlomke za datu konstantu alpha:

# In[10]:


k = m-n
alpha_verizni=[]

x0 = Decimal(alpha)
a0 = Decimal(math.floor(Decimal(x0)))
d0 = Decimal(x0 - a0)
x=[]
x.append(x0)
a=[]
a.append(a0)
alpha_verizni.append(a0)
d=[]
d.append(d0)


for i in range(1,k):
    if d[i-1] <= 0.0000000000000000000000000000000000000001:
        break
    xx = Decimal( 1 / d[i-1])
    x.append(xx)
    aa = Decimal(math.floor(Decimal(xx)))
    a.append(aa)
    alpha_verizni.append(aa)

    dd = Decimal(xx - aa )
    d.append(dd)

last_index = len(a)-1
```

```python
if a[last_index] == 1:
    a[last_index - 1] = a[last_index - 1] + 1
    alpha_verizni[last_index - 1] = alpha_verizni[last_index - 1] + 1
    del a[last_index]
    del alpha_verizni[last_index]

res = "["
for i in range(len(alpha_verizni)):
    res = res + str(alpha_verizni[i])
    if(i==0 and len(alpha_verizni)>1):
        res = res + ";"
    else:
        if i!=len(alpha_verizni)-1:
            res = res + ","
res = res + "]"
print(res)


# Trazimo aproksimacije prve i druge vrste:
#

# In[11]:


for i in range(0, len(all_fractions)):
    fr = all_fractions[i]
    if fr.e2 < e2_min:
        e2_min = fr.e2
        best_second.append(fr)
        if fr.e1 < e1_min:
            e1_min = fr.e1
        continue
    if fr.e1 < e1_min:
        e1_min = fr.e1
        best_first.append(fr)


# Ispis prve vrste:

# In[12]:


print("APROKSIMACIJE PRVE VRSTE:")
for fr in best_first:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))

print("Sortirano:")
sorted_best_first = sorted(best_first, key=lambda fr : fr.e1)
for fr in sorted_best_first:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))


# Ispis druge vrste:

# In[13]:
```

```
print("APROKSIMACIJE DRUGE VRSTE:")
for fr in best_second:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))

print("Sortirano:")
sorted_best_second = sorted(best_second, key=lambda fr : fr.e1)
for fr in sorted_best_second:
    print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) +"  e2 = "+ str(fr.e2_sign))


# Konacan ispis:

# In[14]:


for fr in sorted_all_fr:
    if fr in best_first:
        print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) + "   I")
    elif fr in best_second:
         print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) + "  II")
    else:
         print(fr.printFr() + " " + fr.printVerizni()  + "  e1 = "+
str(fr.e1_sign) + "  N")
```