PYTHON


# Osnove 8, številski sistemi

Python urejevalnik se nahaja na strani:
`https://www.w3resource.com/python-exercises/python-basic-exercises.php#EDITOR`
Naloge Kattis:`https://github.com/minidomo/Kattis`

1. **Trojiški številski sestav** Številski sestavi določajo, kako se zapisujejo števila. Mi uporabljamo desetiški številski sestav. Osnova tega sestava je $b = 10$. Števk, znakov s katerimi zapisujemo števila, je tudi 10. To so `0,1,2,3,4,5,6,7,8,9`. Njihove številske vrednosti so $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. Velja

$$53107_{10} = 5\,10^5 + 3\,10^4 + 1\,10^3 + 0\,10^2 + 7\,10^0.$$

V mestnem zapisu je zelo pomembno, na katerem mestu stoji posamezna števka. Manjkajoče potence števila 10 v razvoju napovemo s števko `0`.

Ogledali si bomo poseben številski sestav. V osnovi bo to trojiški sestav. V običajnem trojiškem številskem sestavu potrebujemo tri števke `0,1,2`, katerih številske vrednosti so enake $0, 1, 2$. Zapis `2101` v tem sestavu predstavlja število

$$2101_3 = 2\,3^3 + 1\,3^2 + 0\,3^1 + 1\,3^0 = 2\,27 + 9 + 1 = 54 + 9 + 1 = 64_{10}.$$

Vendar pa je naš trojiški sistem drugačen. Koeficienti pred potencami števila 3 niso 0, 1 in 2, ampak $-1$, 0 in 1. Števke bomo označili z znaki `-,o,+`, številske vrednosti pa so enake $-1, 0, 1$. Zapis `+00-00` predstavlja število

$$(\texttt{+oo-oo}) \to 3^5 - 3^2 = 234_{10}.$$

Če v zapisu zamenjamo med seboj znaka `-` in `+`, dobimo ustrezen zapis števila, pomnoženega z $-1$.
**Definicija problema** Dano število zapišimo v tem številskem sestavu. Program naj vrne ustrezen niz števk.
**Vhod** Na standardnem vhodu preberemo naravno število.
**Izhod** Na standardni izhod zapišemo niz števk, ki predstavljajo število, zapisano v našem številskem sestavu.
**Primera**

```
Vhod:
1. 32567          2. 666
Izhod:
1. +--oo-oo+--    2. +o-+-oo
```

2. **Splošni številski sestavi** Napišite program, ki bo sprejemal dva celoštevilčna parametra, prvo je baza številskega sestava in drugo je število, ki ga zapišemo v številskem sestavu dane baze. Izpis je v obliki niza števk. Vzeli bomo, da je $baza \leq 16$, števke pa so `"0123456789ABCDEF"` Poleg tega napišite še program ki vam zapis števila v dani bazi pretvori z desetiški zapis.
**Vhod:** Na standardnem vhodu preberemo dve naravni števili `base n`.
**Izhod:** Na standardni izhod zapišemo niz števk, ki predstavljajo število $n$, zapisano v številskem sestavu dane baze *base*.
**Primera**

```
Vhod:
1. 12 32567      2. 13 666
Izhod:
1. 16A1B         2. 3C3
```

3. **Številski sistem Fibonacci**

**Fibonaccijevo zaporedje** *Člen v Fibonaccijevem zaporedju je vsota dveh neposredno predhodnjih členov. Prva dva člena zaporedja pa sta enaka* 1.

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+1} = F_n + F_{n-1}, \quad n > 0.$$

Vsako pozitivno celo število je mogoče zapisati kot vsoto različnih členov Fibonaccijevega zaporedja. Na primer, $10 = 8 + 2$, vsota petega in drugega člena Fibonaccijevega zaporedja.

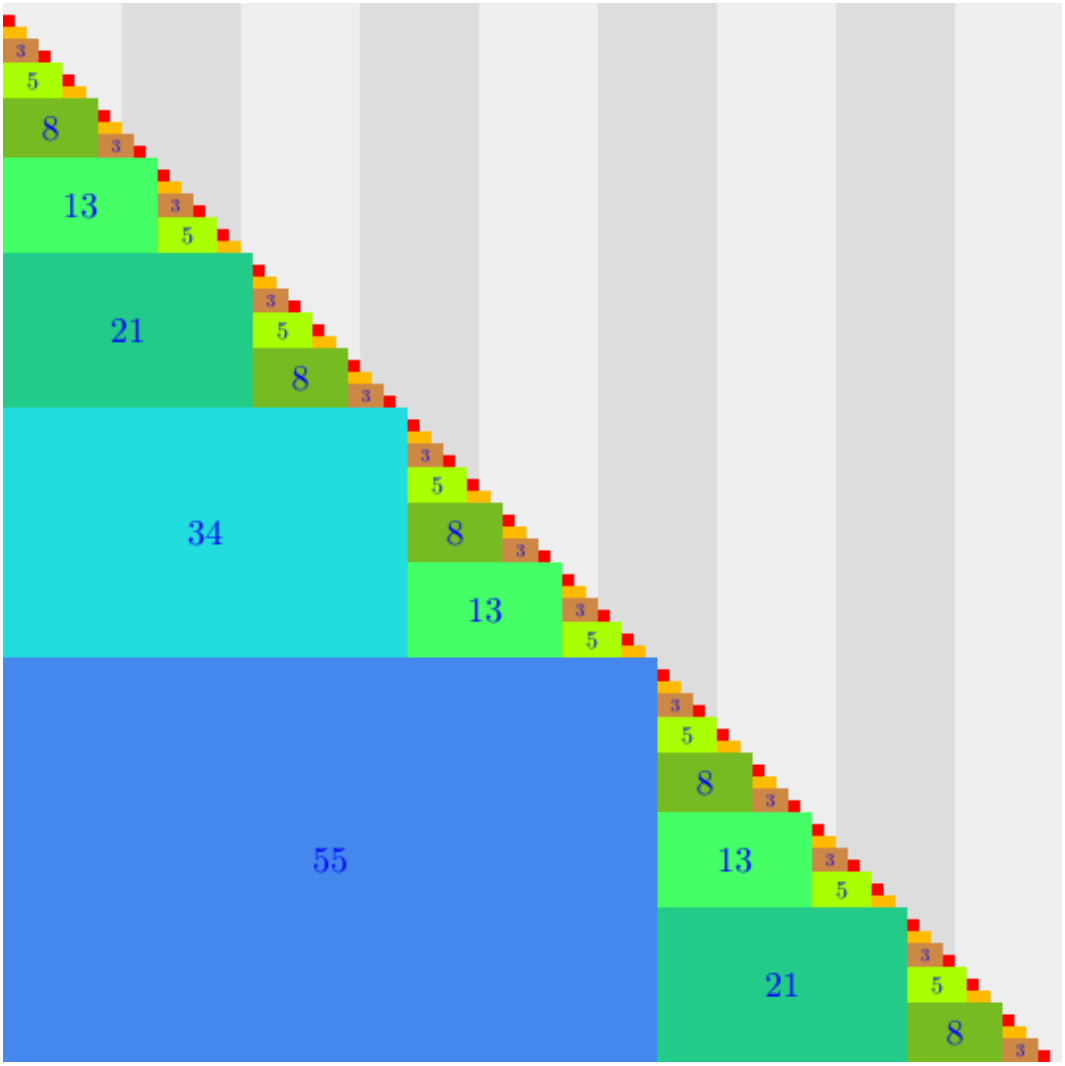
Ta razčlenitev je enolična, če zahtevamo, da dva zaporedna člena Fibonaccijeva zaporedja nista dovoljena. Vsota dveh zaporednih členov Fibonaccijevega zaporedja je naslednji člen Fibonaccijevega zaporedja. Iz tega skledi, da dva zaporedna člena lahko nadomestimo z naslednjim členom.

**Zeckendorfov izrek**
*Vsako pozitivno celo število je mogoče enolično zapisati kot vsoto različnih nesosednjih členov Fibonaccijevega zaporedja.*
**Naloga**
Za dano je naravno število, poiščite predstavitev le-tega kot vsoto nezaporednih členov Fibonaccijevega zaporedja števil.
**Primera**

```
32576 ---> [28657, 2584, 987, 233, 89, 13, 3, 1]
666   ---> [610, 55, 1]
```

**Zeckendorfov algoritm**

a) Naj bo $n$ naravno število na vhodu.
b) Medtem ko je $n > 0$:
   i. Poiščemo največji člen $f$ Fibonaccijevega zaporedja, ki je manjši ali enak $n$. Dodamo število $f$ v seznam.
   ii. Postavimo $n = n - f$ in gremo v 3b.
c) Izpišemo seznam členov Fibonaccijevega zaporedja, ki smo jih nabrali na poti.


3
5
8
3
13
3
5
21
5
8
3
34
3
5
8
3
13
3
5
55
3
5
8
3
13
3
5
21
3
5
8
3

# Rešitve 8

## Trojiški številski sestav

```python
#!/usr/bin/env python3
tri_digit = {'+': 1, 'o': 0, '-': -1}
digit_tri = {1: '+', 0: 'o', -1: '-'}

def to_tri(n):
    res = ''
    while True:
        a = n // 3
        b = n % 3
        if b == 2:
            res += digit_tri[-1]
            a += 1
        else:
            res += digit_tri[b]
        if a == 0:
            return res[::-1]
        n = a

def to_dec(s):
    print(s)
    res = 0
    res = tri_digit[s[0]]
    for x in s[1:]:
        res = res * 3 + tri_digit[x]
    return res

if __name__ == '__main__':
    num = int(input(' num -> '))
    num3 = to_tri(num)
    print(num3)
    print(to_dec(num3))
```

## Splošni številski sestavi

```python
#!/usr/bin/env python3

digits = '0123456789ABCDEF'
digits = dict(enumerate(digits))
nums = {v: k for k, v in digits.items()}

#print(digits, nums)

def dec_to_base(base, num):
    res = []
    while num:
        res.append(num % base)
        num //= base
    return ''.join([digits[x] for x in list(res[::-1])])


def base_to_dec(base, num):
    num = [nums[x] for x in num]
    res = num[0]
    for i in num[1:]:
        res = res * base + i
    return res

if __name__ == '__main__':
    base, num = [int(x) for x in input('base num -> ').split()]
    num1 = dec_to_base(base, num)
    print(num1)
    print(base_to_dec(base, num1))
```

### Fibonacciev številski sestav

```python
#!/usr/bin/env python3

def zekendorf(n):
    res = []
    fib_seq = [0, 1]
    while n > fib_seq[-1]:
        fib_seq.append(fib_seq[-2] + fib_seq[-1])
    index = len(fib_seq) - 1
    fib_dic = dict(zip(fib_seq, range(len(fib_seq))))
    print(fib_seq, fib_dic)
    if fib_seq[index] == n:
        res.append(n)
        return res
    index -= 1
    dn = n - fib_seq[index]
    res.append(fib_seq[index])
    while True:
        while dn < fib_seq[index]:
            index -= 1
        res.append(fib_seq[index])
        if dn == fib_seq[index]:
            return res
        else:
            dn = dn - fib_seq[index]

if __name__ == '__main__':
    n = int(input(' n -> '))
    N = zekendorf(n)
    print(N, sum(N))
```