Rekursioon II

12. Tahvlipraktikum

Kaspar Sarapuu
goots@ut.ee

Rekursiooniskeemid

Eelmises praktikumis vaadeldi lineaarse rekursiooniga funktsioone, kus funktsioon kutsub ennast välja ainult ühe korra.

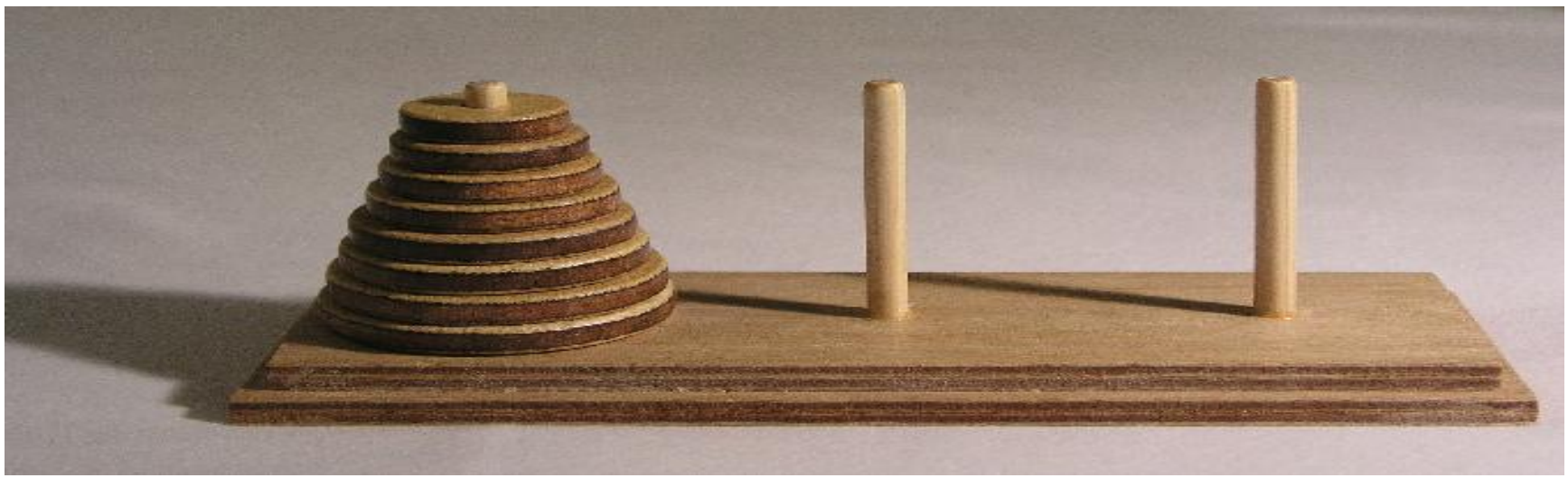
Käesolevas praktikumis uuritakse keerukamaid rekursiooniskeeme, kus funktsioon kutsub ennast välja mitmes kohas.

Hanoi tornid

Mängulaud koosneb kolmest vardast ja nende peale laotud *n* erineva suurusega ketast. Mängu alguses on kettad laotud suuruse järjekorras esimesele vardale.

Mängu ühe käiguga tõstetakse üks ketas teisele vardale, kuid suuremat ketast ei tohi kunagi panna väiksema peale.

Mängu eesmärgiks on laduda kõik kettad kolmandale vardale.



Hanoi tornid 2

Arutlus

Baasjuhtum 1 ketas:

Tõstame ketta kolmandale vardale.

Samm *k* ketast:

1. Laome pealmised *k -1* ketast vahepealsele (teisele) vardale.
2. Tõstame suurima ketta kolmandale vardale.
3. Laome *k -1* väiksemat ketast suure ketta peale.

Hanoi tornid 3

```
def laoTorn(n, kust, kuhu, puhver):
    if n == 1:
        print(kust, " > ", kuhu)
    else:
        laoTorn(n - 1, kust, puhver, kuhu)
        print(kust, " > ", kuhu)
        laoTorn(n - 1, puhver, kuhu, kust)


n = int(input('Ketaste arv: '))
laoTorn(n, 'Esimene', 'Kolmas', 'Teine')
```

Vastastikune rekursioon

```
def even(n):
    if n == 0:
       return True
    else:
       return odd(n - 1)

def odd(n):
    if n == 0:
       return False
    else:
       return even(n - 1)
```

Fibonacci arvud

$$F_1 = 1$$
$$F_2 = 1$$
$$F_k = F_{k-1} + F_{k-2}$$

Fibonacci arvud

```
def fibo(n):
    if n == 0:
        return 0
    if n == 1 or n == 2:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)
```

Fibonacci arvud

```
def fibo(n):
    if n == 0:
        return 0
    if n == 1 or n == 2:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)
```

Mis siin halvasti on?

Fibonacci arvud

```
def fibo(n):
    if n == 0:
        return 0
    if n == 1 or n == 2:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)
```

Mis siin halvasti on?

Jõudlus!

Fibonacci arvud optimeeritud

```
def fibo(n):
    if n == 0:
        return 0
    else:
        return arvuta_fibo(n, 0, 1)

def arvuta_fibo(n, a, b):
    if n == 0:
        return a
    else:
        return arvuta_fibo(n - 1, b, a + b)
```

Ackermanni funktsioon

Naturaalarvuliste m ja n korral:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

http://gfredericks.com/sandbox/arith/ackermann

Ackermanni funktsioon

```
import sys
sys.setrecursionlimit(2000)

def ack(x, y):
    if x > 0:
        if y > 0:
            return ack(x - 1, ack(x, y - 1))
        else:
            return ack(x - 1, 1)
    else:
        return y + 1
```

Ülesandeid

1. Antud on *n* * *m* ruudustik. Kui palju on erinevaid teid ülevalt vasakust nurgast alla paremale nurka kui nõuame, et iga samm viib sihtmärgile lähemale (st. ühe ruudu võrra alla, paremale või diagonaalis alla paremale)?

2. Pythonis on listi summa leidmiseks funktsioon `sum()`. Luua funktsioon, mis leiab listi summa ka juhul, kui list koosneb listidest.

```
>>summa([ 1, 2, [3, 4] ]) => 10
```

3. Koostada rekursiivne funktsioon, mis leiab jada suurima/vähima liikme.

4. Koostada rekursiivne funktsioon, mis leiab jada esimese negatiivse elemendi.

5. Koostada rekursiivne funktsioon, mis kontrollib, kas jadas leidub otsitav element.

6. Koostada rekursiivne protseduur, mis trükib täisarvu kolmestesse gruppidesse jaotatuna. Näiteks: 1'234'567

7. Koostada funktsioon `on_aste(a,b)`, mis tagastab `True` juhul, kui *a* on *b* aste. Näiteks `on_aste(1024,2) => True`.