Programmeerimise süvendatud algkursus ITI0140

2014

Jadaotsing

1. Ma valin arvu 0-10 vahel (nt 4).
2. Kas on 0? Ei.
3. Kas on 1? Ei.
4. Kas on 2? Ei.
5. Kas on 3? Ei.
6. Kas on 4? Jah.

- Et teada saada, et arvu jadas pole, tuleb kõik elemendid üle kontrollida, st sooritada **n** kontrolli.
- Töötab suvalise järjestamata jada peal.
- Keerukus: **O(n)**

Kahendotsing

1. Ma valin arvu 0-100 vahel (nt 72).
2. Kas on 50-st suurem? Jah → 50-100 vahel.
3. Kas on 75-st suurem? Ei → 50-75 vahel.
4. Kas on 62-st suurem? Jah → 62-75 vahel.
5. Kas on 68-st suurem? Jah → 68-75 vahel.
6. Kas on 71-st suurem? Jah → 71-75 vahel.
7. Kas on 73-st suurem? Ei → 71-73 vahel.
8. Kas on 72-st suurem? Jah → 72-73 vahel.
9. Kas on 72? Jah.

- Et teada saada, et arvu jadas pole, kulub **log2(n)** kontrolli.
- Eeldab, et jada on järjestatud (st sortimine on vajalik).
- Keerukus: **O(log2(n))**

Ülesanne

Kasutades etteantud moodulit **tund16gen.py**, kirjutada ise otsingu funktsioonid, mõõta nende töökiirust ja esitada tulemused nii tabeli kui ka graafikuna.

Moodul sisaldab lähteandmete generaatorit. Funktsioon **gimme_my_input** võtab vastu argumentidena genereeritava jada pikkuse ja juhuarvude algväärtustamise seemne ning tagastab paari (kaheelemendilise enniku), kus esimene element on soovitud pikkusega genereeritud jada ja teine jada genereerimiseks kasutatud generaator.

Otsingufunktsioonid peaks vastavast jadast otsima generaatorist tulevaid järgmisi arve. Kui jada pikkus on **n**, siis võiks otsida järgmist **n** generaatorist tulevat juhuslikku arvu. Leidmise fakt pole oluline, kuid katse läbiviimise aeg on.

Ülesanne

Realiseerida tuleks jadaotsing (**0** korda sortida, **n** korda otsida), kahendotsing (**1** kord sortida, **n** korda otsida) ja kahendotsing, mis sordib jada igal otsingul (**n** korda sortida ja otsida). Sortimiseks võib kasutada Pythoni mõnda sisseehitatud algoritmi. Otsingufunktsioonid ei peaks uusi järjendeid looma, st kahendotsingul kasutage ülemise ja alumise raja meeldejätmiseks indekseid.

Pidage meeles, et mõõtmised oleks õiglased iga algoritmi suhtes, siis tuleb nii jada kui generaator algväärtustada enne iga katse sooritamist, seejuures sama juhuarvude algväärtustamise seemnega.

Mõõtmised tuleks teha erinevate jada pikkustega. Kõige mõistlikum oleks kasutada **2** astmeid, st pikkustega **1**, **2**, **4**, **8**, …, **2^m**. Katsetage ise, kui suureks on mõtet **n=2^m** ajada. Graafikule tuleks kanda tulemused, mis näitavad iga algoritmi töötamiseks kuluva aja sõltuvust jada pikkusest. Telgedel kasutage logaritmskaalasid.

tund16gen.py

```
"""
Module to generate lists and random numbers.
"""

import random

def __gimme_a_generator(max_number, seed):
    """
     Generator to use for generating random numbers.
     """
     r = random.Random(seed)
    while True:
        yield r.randrange(1, max_number)

def __gimme_a_list(generator, size):
    """
     Returns a generated list to use for searching.
     """
    return [next(generator) for _ in range(size)]
```

tund16gen.py

```
def gimme_my_input(size, seed):
    """
    Returns a tuple consisting of the generated list and the generator to use.
    """
    gen = __gimme_a_generator(2 * size, seed)
    lst = __gimme_a_list(gen, size)
    return (lst, gen)


def __main():
    inp = gimme_my_input(10, "123456")
    print(inp[0])
    print(next(inp[1]))
    print(next(inp[1]))

if __name__ == "__main__":
    __main()
```