Programmeerimise süvendatud algkursus ITI0140

2015

Teemad

- Simuleerimine, kui probleemi lahendamise meetod
- Monte Carlo meetod
- Pseudo-juhuslikud arvud
- Jõudluse mõõtmine
- Ülesanne

Simuleerimine

- Simuleerimisega puutusime kokku robotite ülesande juures, kus meil oli virtuaalne maailm ja teie kirjutatud robot, mille käitumist me selles maailmas vaatasime
- Simuleerimine pole aga lihtsalt mäng, paljude pärismaailma protsesside modelleerimisel tekib tihti vajadus mingeid olukordi läbi simuleerida
- Tihti pole aga mõtekas (ega ka võimalik) neid olukordi läbi teha päris maailmas
  - see võib kaua aega võtta
  - see võib olla tehniliselt keerukas
  - see võib palju ressursse nõuda
  - olukorrad ei pruugi lihtsasti korratavad olla
- Nii mõnegi probleemi lahendamisel võib piisata ka ligikaudsetest tulemustest
- See ei tähenda, et simuleerimine on lihtne – tihti on need rakendused vägagi keerulised
- Küll aga võib simuleerimine oluliselt kiiremini keerulistele probleemidele lahendamisele uut infot anda

Monte Carlo meetod

- Ühest ja täpset definitsiooni meetodil pole
- Katab laia hulga arvutuslikke algoritme ja põhineb juhusliku valimi kasutamisel mingi arvutusliku tulemi leidmisel
- Kasutatakse tihtipeale matemaatilistes ja füüsikalistes süsteemides
- Arvutustes kasutatakse juhuslike arvude generaatoreid
- On probleeme, mille puhul on vaja tõeliselt juhuslikke arve (näiteks algarvude leidmisel, krüptograafias)
- Tihti piisab ka piisavalt juhuslikest arvudest, siis kasutatakse pseudo-juhuslike arvude generaatoreid
- Sai alguse Los Alamose laboratooriumis '40 aastatel tuumafüüsika arvutuste käigus

Pseudo-juhuslikud arvud

- Pseudo-juhuslike arvude generaatorile antakse tüüpiliselt ette seeme (*seed*)
  - Vaikimisi kasutatakse süsteemset kella
- Seemnest lähtuvalt tekitatakse mingi algoritmi alusel juhuslike arvude jada
- Hea algoritmi korral on tekitatud arvude jada ühtlase jaotuvusega ja näib juhuslik
- Andes ette sama seemneväärtuse, on võimalik juhuslike arvude jada korrata
- https://docs.python.org/3/library/random.html#module-random

```
import random
random.seed("kala")
print([random.randint(0, 15) for x in range(10)])
random.seed("lammas")
print([random.randint(0, 15) for x in range(10)])
random.seed("kala")
print([random.randint(0, 15) for x in range(10)])
```

```
[0, 15, 2, 10, 5, 13, 2, 5, 9, 5]
[3, 1, 1, 4, 9, 2, 10, 0, 11, 5]
[0, 15, 2, 10, 5, 13, 2, 5, 9, 5]
```

Ajavõtt

- Teatud olukordades on oluline mõõta, kaua programm või selle osa aega võtab
- Lihtsal juhul võib selleks kasutada **time.time()** funktsiooni

```
import time

start = time.time()
for _ in range(1000000):
    pass
end = time.time()

print(end - start)
```

```
0.07999992370605469
```

Profileerimine

- Profileerimiseks nimetakse tüüpiliselt rakenduse analüüsi meetodit, millega jälgitakse:
  - operatsioonide sagedust
  - kestvust
  - ressursside kasutamist jne
- Hindamiseks kasutatakse erinevaid statistilisi näitajaid (min, max, avg)
- Aitab leida vigu nii koodis kui algoritmi disainis
- https://docs.python.org/3/library/timeit.html#module-timeit

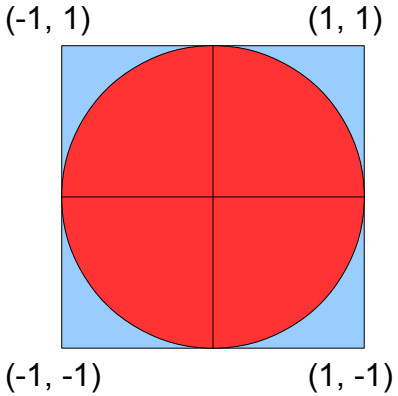
```
import timeit as t

def takes_long():
    for _ in range(100000000):
        pass

print(t.timeit(takes_long, number = 1))
```

```
2.752061813107995
```

π arvutamine Monte Carloga

Selleks tuleb võtta ruudukujuline alus ja sellele joonistada maksimaalse võimaliku suurusega ringikujuline märklaud. Kui loopida seda märki juhuslikult nooltega, siis märki tabavate noolte (ringi sisse) ja mööda minevate noolte (ruudu sees ja väljaspool ringi) suhe on määratud märklaua ja aluse pindala suhtega.

Kui visata **n** noolt ja neist **h** tabab märklauda (ringi sisse), siis **π ≈ 4 * h / n**.



Seda on võimalik simuleerida kasutades ühikringi keskpunktiga koordinaatide alguspunktis. **x** ja **y** koordinaadi leidmiseks võib kasutada kummalgi juhul kasutada avaldist kujul **2*random() - 1**.

x^2 + y^2 <= 1 korral asub leitud punkt ringi sees.

http://en.wikipedia.org/wiki/Monte_Carlo_method#Introduction

Ülesanne

Ülesanne on nähtaval

- https://ained.ttu.ee
- https://courses.cs.ttu.ee/pages/ITI0140