Algoritmid ja andmestruktuurid

- Sissejuhatus kursusesse
- Sissejuhatav näide

Marko Kääramees
TTÜ arvutiteaduse instituut
marko.kaaramees@ttu.ee

Näide: marsruudi planeerimine

Leida kiireim tee punktist A punkti B

- Andmestruktuurid

  Andmed tuleb kuidagi esitada

  – Sõidurajad, võimalikud pöörded, mitmetasandilised teed, ühesuunalised tänavad, liikumiskiiruse erinevus

- Kiire algoritm
  – Piiratud arvutusvõimsus
  – Suur andmemaht (kõik Euroopa teed-tänavad)
  – Arvuti ei vaata kaardile “ülevalt-alla” arvutada tuleb andmebaasi kirjetel, mitte graafilisel pildil

Arvutuslikult keerukad probleemid

- Arvutuslik keerukus ilmneb, kui
  - tuleb töödelda suuri andmekoguseid
  - vastata tuleb ajapiirangu raames (*online*)
  - tihti seotud optimiseerimisega (leia parim)

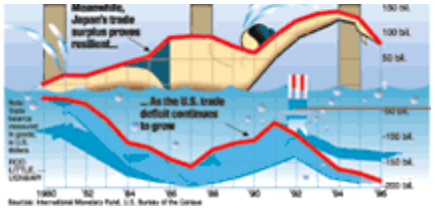
- Näiteid
  - Leia odavaim lennupilet punktist A punkti B
  - Leia ajapiirangutele ja töötajate/seadmete oskustele vastav tööde teostamise plaan
  - Leia kõik vastamata päringud
  - Leia testide komplekt, mis testiks kõiki nõudeid

Keeruka probleemi lahendamine

Probleem

Andmed

Algoritm



Andmestruktuur

Programm

Tulemus

Mis kasu on algoritmidest ja andmestruktuuridest?

- Projekt enne ehitamist
- Algoritmid (*kuidas ehitada*)
  - probleemi taandamine teadaolevale algoritmile
  - efektiivse algoritmi loomine
- Andmestruktuurid (*millest ehitada*)
  - abstraktne, arusaadav (ja standardne) liides
  - andmete efektiivne hoidmine ja efektiivsed operatsioonid vastavalt algoritmi vajadustele

Mis on andmestruktuur?

Liides

Implementatsioon

Kursuse eesmärgid

Õpime tundma:

- erinevaid algoritmide loomise paradigmasid:
  - *jaga ja valitse* algoritmid
  - *ahned* algoritmid
  - *tagasivõtmisega* algoritmid
  - *hargne ja kärbi* algoritmid
  - *pseudoheuristilised meetodid*
- erinevate andmestruktuuride kasutamist:

  listid, puud, graafid, paisksalvestus, järjekorrad

  ja nendel töötavaid tuntumaid algoritme
- elementaarset keerukusteooriat ja keerukuse hindamise meetodeid

Fibonacci arvud probleem

- Leonardo Pisast (aka Fibonacci) huvitus mitmetest matemaatilistest probleemidest, sealhulgas populatsioonide dünaamikast.

- Akadeemiliste jäneste populatsioon:
  - igal jänesepaaril on igal aastal kaks järeltulijat
  - jäneste lapsed ei saa lapsi esimesel eluaastal
  - jänesed ei sure kunagi

Kui palju on jäneseid *n* aasta pärast?

Fibonacci arvud rekursiivne funktsioon

- F(*n*) - jänesepaaride arv aastal *n*

  F(1) = 1 Aadam ja Eeva - kõik algab ühest paarist

  F(2) = 1 esimeste jäneste muretu lapsepõlv

  F(3) = 2 vallatu noorus ja esimene paar järeltulijaid

  F(4) = 3 keskea rõõmud ja teine paar järeltulijaid

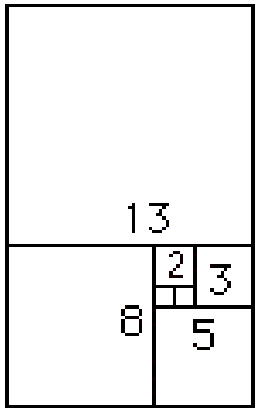
  F(5) = 5 küpsus ja esimesed lapselapsed

  . . .

- Üldkujul F(n) = F(n-1) + F(n-2):
  - F(n-1) kõik senised paarid on elus
  - F(n-2) iga vähemalt kahe aasta vanuse paari kohta tuleb uus paar

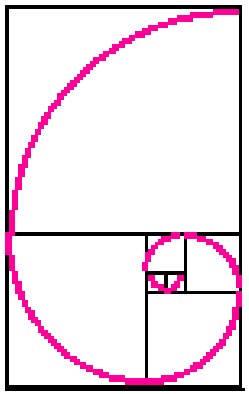
Fibonacci arvud

1 1 2 3 5 8 13 21 34 55 ...



- Fibonacci kahe järgneva arvu suhe ligineb kuldlõikele

$$\varphi = \lim_{n\to\infty} F(n+1)/F(n)$$



- Looduses esineb mitmeid Fibonacci arvude seeriale vastavaid nähtusi

Bert Myers ~ X-ray: Nautilus Shell

Fibonacci arvud
algoritm 1 (rekursiivne algoritm)

```
int fib(int n)
{
  if( n <= 2)  return 1
  else     return fib(n-1) + fib(n-2)
}
```

- Tegemist on rekursiivse algoritmiga
  - lõpetamistingimuse täitmisel algoritm peatub
  - muul juhul kutsub funktsioon välja iseend muudetud argumendiga

Fibonacci arvud algoritmi analüüs

- Kui palju aega võtab selle algoritmi täitmine?
- Kuidas mõõta algoritmi täitmise aega?
  - sekundites? üha uued protsessorid
  - protsessori käskudes? erinevad kompilaatorid
  - programmi ridades? erinevad kodeerimisstiilid
  - elementaartehetes (+,*,<) tehetel erinev keerukus

  selles näites kasutame mõõduks ridade arvu
- Kuidas sõltub täitmise aeg sisendparameetri(te) suurusest?

Fibonacci arvud algoritmi analüüs

```
int fib(int n)
{ if( n <= 2 )   return 1
  else return fib(n-1)+fib(n-2)
}
```

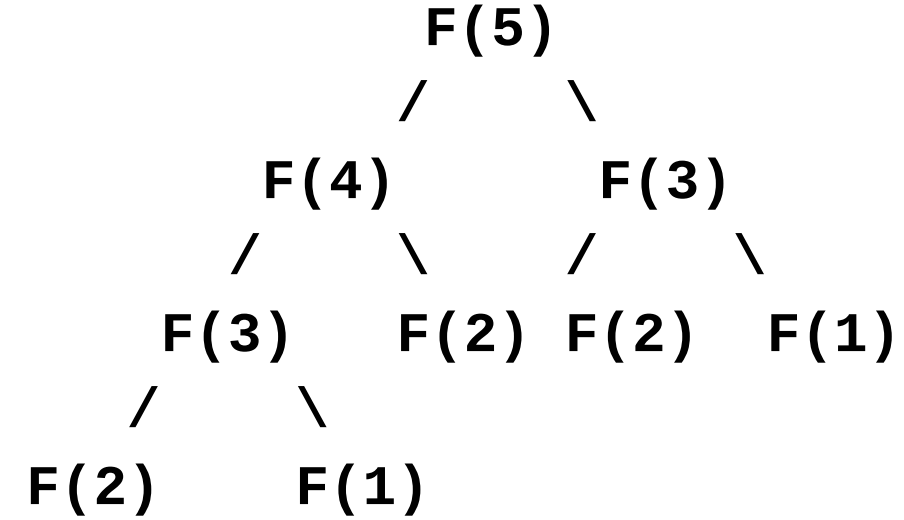
- Iga *fib* funktsiooni väljakutse on kas 1 või 2 rida
  - n <= 2 korral: 1 rida
  - n = 3 korral 2 rida + 1 rida kummagi uue väljakutse kohta: 4
  - n = 4 korral: 2 + 4 + 1 = 7
  - n = 5 korral: 2 + 7 + 4 = 13
- Keerukuse rekurrentne võrrand

  ridu(n) = 2 + ridu(n-1) + ridu(n-2)
- Read paljunevad nagu jänesed

Fibonacci arvud algoritmi analüüs

ridu(n) = 2 + ridu(n-1) + ridu(n-2)

- Püüame lahendada selle võrrandi F(n) suhtes



Sellises puus on iga n korral

- F(*n*) lehte (F(1), F(2))
- F(*n*) - 1 sisemist sõlme (F(k), k > 2)

Fibonacci arvud algoritmi analüüs

- F(n) lehte
  = F(n) rida programmi
- F(n) - 1 sisemist sõlme
  = 2(F(n) - 1) rida programmi
- Kokku 3 F(n) - 2 rida programmi

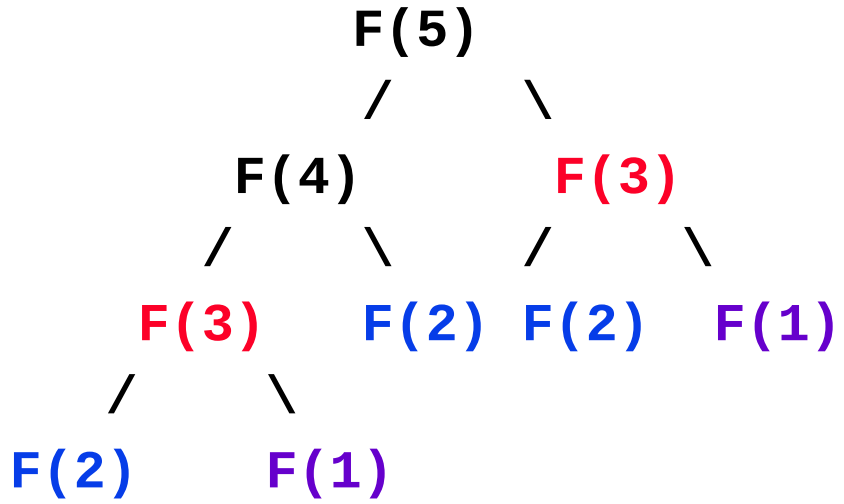
ridu(n) = 3 F(n) -2

ridu(5) = 3 * 5 - 2 = 13

ridu(45) > 1000000000

Fibonacci arvud parem algoritm

Algoritmi aegluse põhjus -
lahendame samu alamprobleeme korduvalt



- Alustame arvutamist väiksematest argumentidest, peame tulemused meeles ja arvutame järjest tulemusi suurematele argumentidele

Fibonacci arvud – algoritm 2 dünaamiline planeerimine

```
int fib(int n)
{
  int f[n+1];
  f[1] = f[2] = 1;
  for (int i = 3; i <= n; i++)
    f[i] = f[i-1] + f[i-2];
  return f[n];
}
```

1	1
2	1
3	2
4	3
5	5
6	8
7	13

- Tegemist on iteratiivse algoritmiga
  korduvad arvutused tehakse tsüklis

Fibonacci arvud - algoritmi 2 analüüs

- Iteratiivse programmi puhul peame arvutama kokku kui mitu korda mingit rida täidetakse:
  - kolm rida täidetakse alati
  - tsükli esimest rida täidetakse n-1 korda
  - tsükli teist rida täidetakse n-2 korda
  - kokku täidetakse :

  ridu(n) = 3 + (n-1) + (n-2) = 2n rida

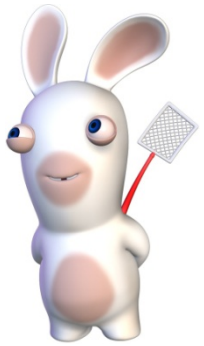
  ridu(45) = 90 (üle 10 miljoni korra kiiremini)
- Mahuline keerukus
  - iga argumendi väärtus´jaoks üks koht massiivis

  size(n) = n

Fibonacci arvud – algoritm 3 (iteratiivne algoritm)

- Eelnevat algoritmi saab veelgi parandada mäluvajaduse keerukuse osas
- Iga samm tsüklis vajab ainult kahte eelnevat Fibonacci arvu väärtust. Massiivi asemel võime kasutada kahte muutujat.

```
int fib(int n) {
  int a = 1, b = 1, c;
  for (int i = 3; i <= n; i++) {
      c = a + b;
      a = b;
      b = c;
  }
  return a;
}
```



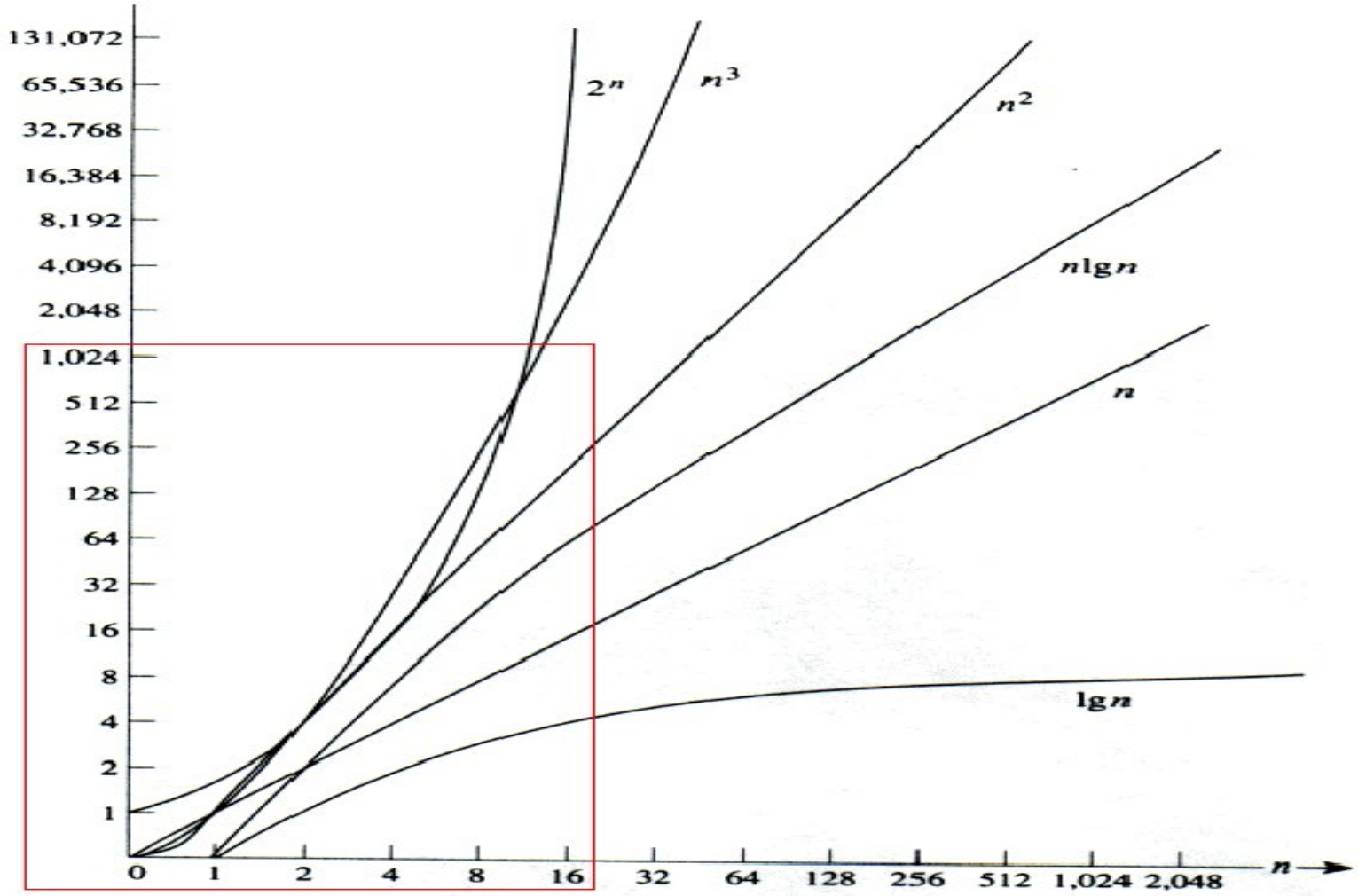
ridu(n) = 2 + (n-1) + 3(n-2) = 4n - 5

Hoiatus: algoritm ei tööta korrektselt, leidke viga!

Järeldused

- Sama probleemi saab lahendada mitme algoritmiga
- Algoritmi idee mängib keeruliste probleemide lahendamisel väga olulist rolli
  - tihti annab hea algoritm palju suuremat võitu kui kiire arvuti, hea kompilaator või kaval viitade aritmeetika
  - mõnikord ei aita isegi parim algoritm
- Keerulised algoritmid kasutavad vahetulemuste hoidmiseks ja nendega opereerimiseks andmestruktuure
- Keerukuse analüüs annab aimu algoritmi headusest ja parema algoritmi olemasolust
- Kiirust saab tihti osta suurema mälukasutuse hinnaga

Erinevad keerukusklassid



Andmete esitamine

- Andmed esitavad tihti mingi nähtuse mudelit
  - Mudel on lihtsustus –
    N: Teedevõrk graafina
  - Esitatakse ainult ülesande lahendamiseks vajalikud parameetrid
    N: teelõigu pikkus, suund aga mitte laius, tõusunurk jms
  - Andmetel on struktuur
    N: Kaart koosneb teelõikudest ja nende ühendustest
- Arvud arvutis - tundub lihtne
  - Kuidas esitada arvutis π?

Täisarvude esitamine

- Täisarvud (int, long)
  - 2-complement esitus
  - Ületäitumine – *overflow*

127	0111 1111
1	0000 0001
0	0000 0000
-1	1111 1111
-2	1111 1110
-128	1000 0000

Ariane 5 rakett plahvatas 40 sekundit peale starti ületäitumise vea tõttu

Ujukomaarvude esitamine

- Ujukomaarvud (float, double)
  - Mitteintuitiivsed probleemid täpsusega
  - Vead võivad kuhjuda

$(-1)^s \times c \times b^q$

$0.85 = -1^{*}0 \times (1+1/2+1/4) \times 2^{-1}$

$0.1 = -1^{*}0 \times (1+1/2+1/16+1/32+1/256+\ldots) \times 2^{-4}$

(double)0.1 → 0.1000000149011612

(0.1+0.1+0.1) != 0.3

Patriot rakett eksis 1991 sihtmärgiga ja tappis 26 sõdurit, kuna aega mõõdeti 0.1 sekundi kaupa.

Suured ja täpsed arvud

- BigInteger
  - Kuitahes täpsed täisarvud
- BigDecimal
  - Eksponent alusel 10
  - Kuitahes palju komakohti

$$\text{BigInteger} \times 10^{q}$$

Algoritmid ja andmestruktuurid

- Kursuse korraldus

Kursuse korraldus

- Loeng
  - esmaspäeviti kell 17.45
- Harjutustunnid)
  - paaris reedel kell 12 ja 14
  - ülesannete lahenduste ülevaatamine ja lahendamine
- Praktikumid
  - paaris reedel kell 8, 10 ja 14
  - tunniülesande lahendamine ning esitamine
  - programmeerimisülesannete kaitsmine
- ained.ttu.ee (Moodle)
  - Materjalid, ülesanded, tulemused
  - Online ülesannete esitamine

Programmeerimisülesanded

- Tuleb implementeerida Javas
- Esitatakse git repositooriumi kaudu
- Kontrollitakse ja hinnatakse automaattestriga nii stiili, korrektsuse kui plagieerimise osas

- Tunniülesanded
  - Antakse praktikumis ja täispunktid, kui lahendatakse praktikumi ajal, hiljem 3 päeva jooksul 50%
- Suuremad programmeerimistööd
  - 3 ülesannet
  - Teostamiseks on aega 2-3 nädalat
  - Tulemus tuleb kaitsta praktikumi juhendajale või saada hinnang *peer review* vormis teistelt üliõpilastelt

Hindamine

- Semestri keskel on kontrolltöö ja lõpus kirjalik eksam
  - kasutada võib paberkandjal materjale
  - tuleb demonstreerida oma arusaamist teemast ja oskust lahendada ülesandeid ning luua algoritme
- Koondhinne
  - 10 punkti *online* ülesannete eest
  - 16 punkti praktikumide tunniülesannete eest
  - 30 punkti programmeerimistööde eest
  - 15 punkti kontrolltöö
  - 30 punkti eksam
  - mõned võimalused saada lisapunkte

  Lõpphinne: 5: 90+ punkti, 4: 80+ punkti, 3: 70+ punkti jne

Mängureeglid

Kontrolltööde, eksamiülesannete lahenduste ja programmide kopeerimine on **plagieerimine**

IT teaduskonnas on kehtestatud kord plagiaatide käsitlemiseks

Kontaktinfo ja materjalid

- loengud ja harjutused: Marko Kääramees
- praktikumid: Evelin Halling, Priit Trink, Marko Kääramees
- Kommunikatsioon
  - Küsimused, mille vastus võib huvitada veel kedagi eelistatult ained.ttu.ee kursuse foorumisse
  - Teated foorumi kaudu
  - Keerulisemad küsimused peale loengut, praktikumis
  - e-mail: algoritmid@cs.ttu.ee, algoritmid-prax@cs.ttu.ee

Veeb ja Moodle

- Koduleht: https://courses.cs.ttu.ee/pages/ITI0050

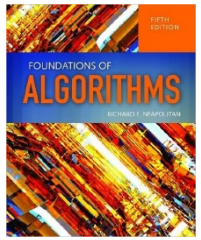
  Viit on olemas ka ainekavas
- Materjalid, tulemused, ülesanded ja *on-line* tegevused keskkonnas ained.ttu.ee Moodle serveris.
- Logige Moodle serverisse oma TTÜ Uni-ID kontoga
  - kasutajanimi@ttu.ee
  - Kasutajanime teadasaamiseks ja salasõna vahetamiseks logige ID kaardiga sisse pass.ttu.ee
- Registreerige kursusele kasutades registreerimiskoodi
  - IAPB51, 52 - praktikum reedel kell 8: algoritm-R8
  - IAPB53, 54 - praktikum reedel kell 10: algoritm-R10
  - IAPB55 - praktikum reedel kell 14: algoritm-R14

Kirjandus

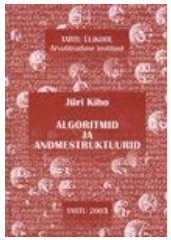
- Kursuse põhiõpik:

  Foundations of Algorithms.

  Kumarss Naimipour.



- Algoritmid ja andmestruktuurid, Jüri Kiho



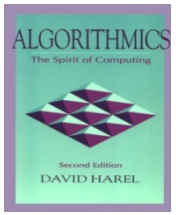
- A Practical Introduction to Data Structures and Algorithm Analysis

  Clifford A. Shaffer

- Kursuse lisalugemist:

  Algorithmics: The Spirit of Computing.

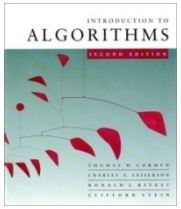
  David Harel



- Põhjalikum käsiraamat

  Introduction to Algorithms

  T. H. Cormen, C. E. Leiserson, R. L. Rivest



Miinimumnõuded kursuse läbimiseks

- Algoritmid
  - suudate kirja panna mittetriviaalse ja efektiivse algoritmi mingi probleemi lahendamiseks

    Leida sorteerimata nimekirjast suuruselt *k*-s element
- Andmestruktuurid
  - suudate valida lahendusalgoritmi jaoks sobiva andmestruktuuri

    Millal kasutada Java *ArrayList*-i või *TreeMap*-i
- Analüüs
  - algoritmile peale vaadates suudate hinnata selle keerukusklassi

    parem/halvem kui $O(n^2)$

Kuidas edasi?

- Registreerige end kursusele
- Praktikumid, harjutused ja *online* ülesanded algavad teisest nädalast