Algoritmid ja andmestruktuurid

- Asümptootiline keerukus, keerukusklassid
- O notatsioon
- Iteratiivsete algoritmide keerukuse analüüs
- Vajalikke põhimõisteid matemaatikast

Hindamine

- 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
- Eksami eeldus
  - 50% online ülesannetest
  - 50% praktikumi tunniülesannetest
  - 3 programmeerimistööd kaitstud positiivse tulemusega
  - 50% kontrolltööst

Töökorraldus sel nädalal

- Registreerige ained.ttu.ee keskkonda, kui ei ole seda veel teinud
  - IAPB51, 52 - praktikum reedel kell 8: algoritm-R8
  - IAPB53, 54 - praktikum reedel kell 10: algoritm-R10
  - IAPB55 - praktikum reedel kell 14: algoritm-R14
- Homme ilmuvad ülesanded harjutustunniks, mida kodus ise enne harjutustundi lahendada
- Tuleb esimene komplekt *online* ülesandeid, tähtaeg järgmise nädala lõpus.
- Praktikumid ja harjutustunnid sel reedel

Kokkuvõte eelmisest loengust

- Sama probleemi saab tavaliselt lahendada mitme erineva algoritmiga
- Algoritmi idee mängib keeruliste probleemide lahendamisel väga olulist rolli
  - tihti annab hea algoritm palju suuremat võitu kui kiire arvuti, hea kompilaator või kaval pointeraritmeetika
  - 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

Algoritmide keerukus

- Algoritmi keerukus on põhioperatsiooni(de) arvu sõltuvusfunktsioon K(*n*) sisendi(te) suurusest *n*.
- *Põhioperatsioon* ei ole üheselt defineeritav

  Midagi mis on riistvaras tehtav piiratud arvu sammudega

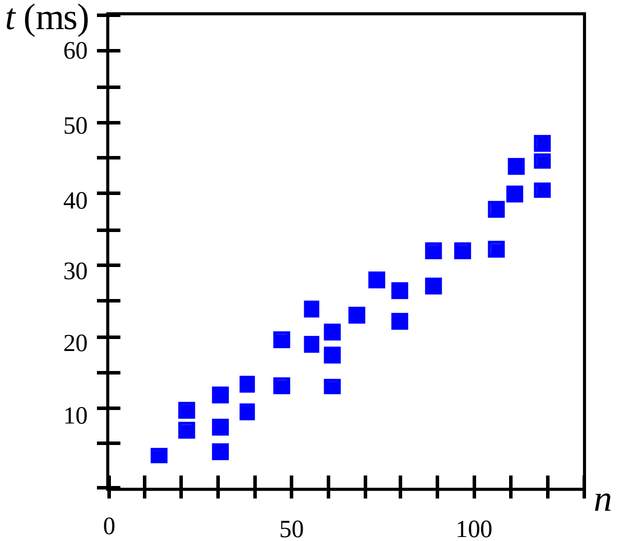
  - aritmeetika tehe, võrdlus, omistus
  - vahel valitakse üks põhioperatsioon ja loetaks selle arvu, näiteks tsüklitingimuse võrdlus
  - mõnikord kasutatakse ka ridade arvu
- *Sisendi suurus* võib olla defineeritud erinevalt
  - Sisendandmete maht (massiivi, faili, andmebaasi suurus)
  - Sisendparameetri väärtus
  - Sisendparameetri suurus (bittide/baitide arv)
- Asümptootiline keerukus ei sõltu põhioperatsiooni valikust

Algoritmi keerukuse mõõtmine

Kuidas mõõta algoritmi täitmise aega?

1. variant: eksperiment

- Kirjutame programmi, mis realiseerib algoritmi
- Laseme algoritmil töötada erineva suuruse ja sisuga andmetega
- Mõõdame programmis täitmise aega. Näiteks Java-s kasutades klassi System.currentTimeMillis().



Eksperimentaalse lähenemise puudused

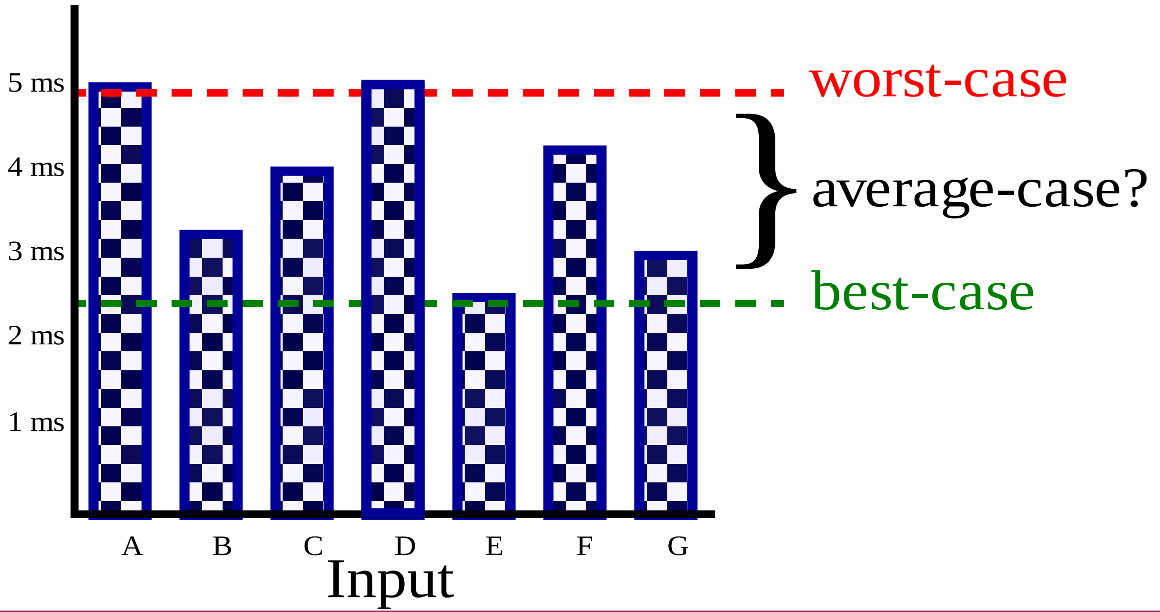
- Algoritmi täitmise aja teadasaamiseks tuleb see kodeerida (programm kirjutada) ja seda testida
- Eksperimente saab teha ainult lõplikul ja üsna piiratud hulgal sisendandmetel. See ei pruugi öelda palju muude võimalike sisendandmete kohta.
- Algoritme tuleb võrrelda samas tark- ja riistvara keskkonnas.

Üldisem analüüsi metoodika

- Vaatame **üldist metoodikat**, mis erinevalt eksperimentaalsest lähenemisest:
  - kasutab algoritmi abstraktset kõrgema taseme esitust, mitte selle realisatsiooni programmina
  - arvestab kõikvõimalikke sisendandmeid
  - võimaldab hinnata algoritmi efektiivsust sõltumatult tark- ja riistvaraplatvormist

Erinevad keerukuskriteeriumid

- Algoritm võib töötada erinevatel sisendandmetel erineva kiirusega
- Keskmise juhu keerukuse leidmine võib olla raske. Seega räägitakse tavaliselt halvima juhu keerukusest.
- Mitmetes kriitilistes rakendustes (lennujuhtimine, meditsiin, IP marsruutimine) omab halvima juhu teadmine olulist tähtsust



Erinevad keerukuskriteeriumid

- Keerukuskriteerium sõltuvalt sisendi valikust
  - Halvima juhu keerukus *W(n)*
  - Keskmine keerukus *A(n)*
  - Parima juhu keerukus *B(n)*
  - Vältimatu keerukus *T(n)*
    - Sõltub ainult sisendi suurusest, mitte sisendi väärtustest
- Mäluvajaduse keerukus
- Teatud juhtudel on oluline ajalise ja mäluvajduse keerukuse korrutis

! Parim juht pole mitte väikseim vaid algoritmi jaoks lihtsaim sisend sama suurusega sisendite seast

Järjestikotsing (list, sortimata massiiv)

- Valime põhioperatsiooniks
  `if (y == x)`
- Halvima juhu keerukus

  $W(n) = n$
- Parima juhu keerukus

  $B(n) = 1$
- Keskmine keerukus

  $A(n) = (n + 1) / 2$
  Kui x on listis

  $A(n) = p(n+1)/2 + (1-p)\ n$

  Kui x on listis tõenäosusega p
- Vältimatu keerukus puudub

```
sequential search(list L,item x)
{
  for y in list L
    if (y == x)
      return y
  return no match
}
```

Asümptootiline keerukus

- **Eesmärk**: lihtsustada analüüsi, jättes ebaolulised detailid arvestamata (analoogselt ümardamisele 1,000,001≈1,000,000)
- Tahame öelda umbes nii

  $3n^2 \approx n^2$

  $3\ F(n) - 2 \approx F(n)$

- Kasutame suure O notatsiooni

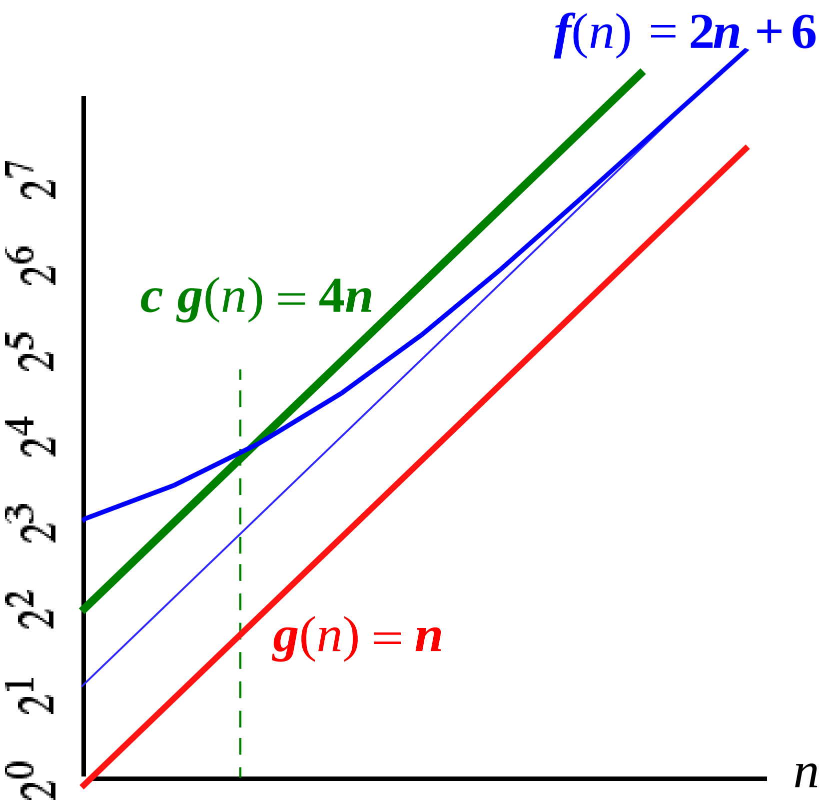
$3n^2$	on $O(n^2)$
$3\ F(n) - 2$	on $O(F(n))$
$4687\ n$	on $O(n)$
$1{,}76 \cdot 10^{25}$	on $O(1)$

Asümptootiline keerukus

Asümtootiline keerukus väljendab põhioperatsioonide sõltuvust sisendi suurusest, kui see kasvab piiramatult

Funktsioonide ***f(n)*** ja ***g(n)*** jaoks on olemas konstandid ***c*** ja *N,* nii et:

$f(n) \leq c\, g(n)$, kui $n \geq N$



järeldus:

2n+6 is O(n).

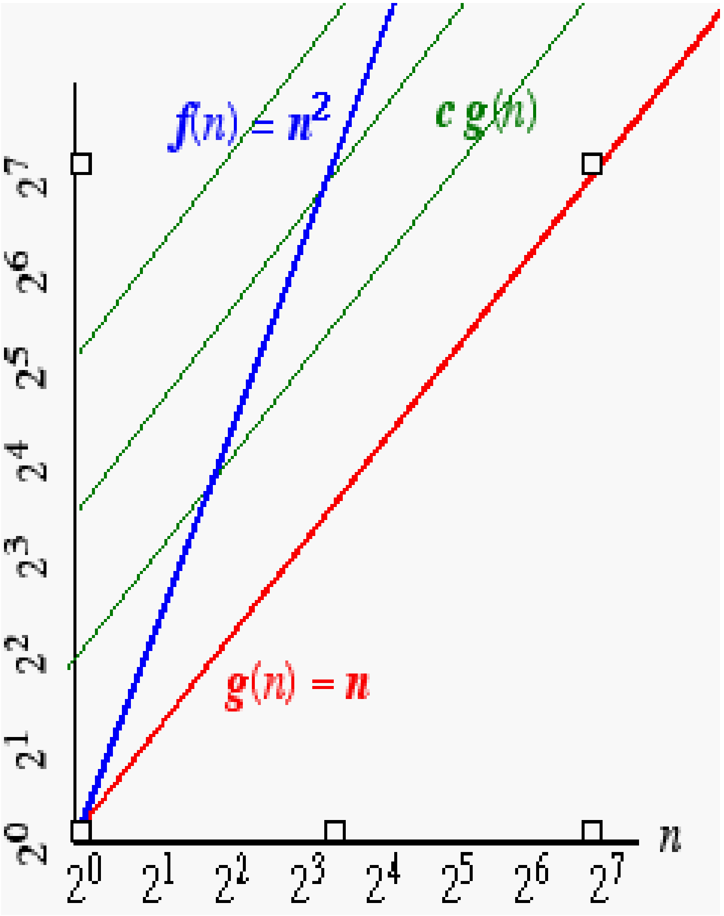
Asümptootiline keerukus

Teisest küljest…

n^2 ei ole $O(n)$ kuna pole konstante c ja N, nii et:

$$n^2 \le cn, \text{ kui } n \ge N$$

(Ükskõik kui suur c valida, ikka on olemas mingi n, nii et $n^2 > cn$)



Üldine keerukusmeetrika, O-notatsioon

Formaalselt:

– *O(g(n))* on funktsioonide ***f(n)*** hulk*,* nii et

$$f(n) < c\, g(n)$$

mingite konstantide, *c > 0*, ja *n > N* korral

st küllalt suure *n* korral

– alternatiivselt võib defineerida, et leidub konstant *c*, mille korral kehtib piirväärtus

***g* on *f* ülemine raja**

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$$

st piirväärtus ei ole ∞

– O(*g*) on funktsioonide hulk, mis ei kasva kiiremini kui *g*

http://en.wikipedia.org/wiki/Big_O_notation

Keerukusmõõdud O, Ω, Θ

- O(*g*)
  - funktsioonide hulk, mis ei kasva kiiremini kui *g*.
- Paar lisanotatsiooni
- Ω(*g*)
- - funktsioonide hulk, mis ei kasva aeglasemalt kui *g*
- - funktsioonide ***f*(*n*)** hulk, nii et

$$f(n) > c\, g(n)$$

mingite konstantide, *c* > *0*, ja *n* > *N* korral

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} > 0$$

g **on** *f*
alumine raja

Keerukusmõõdud O, Ω, Θ

- O(*g*)
  - funktsioonide hulk, mis ei kasva kiiremini kui *g.*
- Paar lisanotatsiooni
- $\Omega(g)$
  - funktsioonide ***f(n)*** hulk, nii et

$$f(n) > c\, g(n)$$

mingite konstantide, *c* > *0*, ja *n* > *N* korral

- $\Theta(g) = O(g) \cap \Omega(g)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$$

Funktsioonide hulk, mis kasvab sama kiirusega kui *g*

Keerukusmõõdud o, ω

- o(*g*)

  funktsioonide $\boldsymbol{f(n)}$ hulk, mis kasvavad aeglasemalt kui *g.*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

- ω(*g*)

  funktsioonide $\boldsymbol{f(n)}$ hulk, mis kasvavad kiiremini kui *g.*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Keerukus kui funktsioonide võrdlus

- Keerukusmõõtusid võib ette kujutada kui võrdlusi funktsioonide vahel

$f(n) \in O(g(n))$	$f \leq g$
$f(n) \in \Omega(g(n))$	$f \geq g$
$f(n) \in \Theta(g(n))$	$f = g$
$f(n) \in o(g(n))$	$f < g$
$f(n) \in \omega(g(n))$	$f > g$

Keerukusmõõt ja keerukuskriteerium

Keerukusmõõt (O, Ω, Θ) ja keerukuskriteerum (parim, halvim, keskmine) on erinevad asjad

Järjestikotsingu halvima juhu keerukuse ülemine raja on lineaarne funktsioon

$W(n) \in O(n)$

Keerukus – põhjuse-tagajärje seos

- Keerukuse hindamine võimaldab mõista mida ühe või teise algoritmi ja operatsiooni kasutamine maksma läheb

Algoritmi analüüsimine - jada

- Lihtoperatsioonid (omistamine, tingimuse võrdlemine)
  - O(1) keerukus ei sõltu *n*-st

- Lihtne käskude järgnevus

  s_1; s_2; …. ; s_k
  - O(1) kui s_i keerukus ei sõltu *n*-st (s on O(1)) ja *k* on konstant
  - Üldjuhul käskude keerutused liituvad
    $O(s_1) + O(s_2) + \ldots + O(s_k) = \max_i(O(s_i))$

- Mitterekursiivse funktsiooni väljakutse

  s_1; f(args); s_2

  $O(s_1) + O(f()) + O(s_2)$

Algoritmi analüüsimine - valik

- Valiku alternatiivide keerukused liituvad

```
if(tingimus)   s1;
else           s2;
```

  - ajaline keerukus on $O(s_1) + O(s_2)$ ehk $\max(O(s_1), O(s_2))$
- Kui valikute tõenäosused on teada, saab arvutada täpsemalt

```
x = random(0..99)
if(x<10)
  for(i=0; i<n; i++) a=a+b
else
  a=a/2;
```

See osa on *O(n)*

  - ajaline keerukus on $0.1\ O(n) + 0.9\ O(1)$ ehk $O(n)$

Algoritmi analüüsimine - tsükkel

- Tsükli keerukus: korduste arv korda tsükli keha keerukus

- Lihtne tsükkel, mis sõltub sisendi suurusest

```
for(i=0;i<n;i++)
    s;
```

kus s keerukus on O(s)

  - ajaline keerukus on n O(s) ehk O(*n*), *kui* O(s) = O(1)

- Üksteises sisalduvad sisendist sõltuvad tsüklid

```
for(j=0;j<n;j++)
    for(i=0;i<n;i++)
        s;
```

See osa on *O(n)*

  - ajaline keerukus: *n * O(n) * O(s)* ehk $O(n^2)$, kui O(s) = O(1)

Algoritmi analüüsimine - tsükkel

- Mitmekordse tsükli keerukus:
  Kõigi tsüklite korduste arv kokku korda tsükli keha keerukus
- Tsükli indeks sõltub välise tsükli muutujast

```
for(j=0;j<n;j++)
   for(k=0;k<j;k++)
     s;
```

  - Sisemist tsüklit täidetakse
    - 1, 2, 3, …., n korda

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

  - Keerukus $O(n(n+1)/2) * O(s) = O(n^2)$, kui $O(s) = O(1)$

Algoritmi analüüsimine

- Tsükli indeks varieerub eksponentsiaalselt

```
h = 1;
while ( h <= n ) {
     s;
     h = 2 * h;
     }
```

  - h saab väärtused 1, 2, 4, … kuni ületab *n*
  - tehakse $1 + \lfloor \log_2 n \rfloor$ iteratsiooni
  - keerukus on $O(\log n)$

```
for(int i=n; i>1; i = i/2)
  s;
```

Algoritmi analüüsimine

- Algoritmi töö sõltub mitmest sisendist

```
for(i=0;i<n;i++)
   for(j=0;j<m;j++)
      s;
```

  - keerukus väljendatakse mitme parameetri kaudu $O(n \cdot m)$

O notatsiooni omadusi

- Konstantseid kordajaid võib ignoreerida
  - $\forall\ k > 0,\ k \cdot f$ on $O(f)$
- Kõrgemad astmed kasvavad kiiremini
  - n^r on $O(n^s)$, kui $0 \leq r \leq s$
- Kiiremini kasvav liidetav määrab summa kiiruse
  - Kui f on $O(g)$, siis $f + g$ on $O(g)$

    näiteks $an^4 + bn^3$ on $O(n^4)$
- Polünoomi kasvu määrab pealiige
  - Kui f on d astme polünoom,
    siis f on $O(n^d)$

O notatsiooni omadusi

- *f* on *O(g)* on transitiivne
  - Kui f on $O(g)$ ja g on $O(h)$, siis f on $O(h)$
- Ülemiste rajade korrutis on korrutise ülemine raja
  - Kui f on $O(g)$ ja h on $O(r)$, siis $f \cdot h$ on $O(g \cdot r)$
- Exponentfunktsioonid kasvavad kiiremini kui astmed
  - n^k on $O(b^n)$ $\forall$ $b > 1$ ja $k \geq 0$
    näiteks n^{20} *on* $O(1.05^n)$
- Logaritmfunktsioonid kasvavad aeglasemalt kui astmed
  - $\log_b n$ on $O(n^k)$ $\forall$ $b > 1$ ja $k > 0$
    näiteks $\log_2 n$ *on* $O(n^{0.5})$

O notatsiooni omadusi

- Kõik logaritmid kasvavad sama kiirusega
  - $\log_b n$ on $O(\log_d n)$ $\forall$ $b, d > 1$

O notatsiooni omadusi

- Kõik logaritmid kasvavad sama kiirusega
  - $\log_b n$ on $O(\log_d n)$ $\forall$ $b, d > 1$

- n r^{th} astmete summa kasvab $(r+1)^{th}$ astmes

$$\sum_{k=1}^{n} k^r \quad \text{on} \quad \Theta(n^{r+1})$$

$$\textbf{\textit{näiteks}} \quad \sum_{k=1}^{n} i = \frac{n(n+1)}{2} \quad \text{on} \quad \Theta(n^2)$$

Kokkuvõte O, Ω, Θ

- O, Ω, Θ on funktsioonide hulgad
  - $O(f)$ hulga ülemine raja
  - $\Omega(f)$ hulga alumine raja
  - $\Theta(f)$ hulga täpne raja

- Abstraheerib konstantse kordaja ja väiksema keerukusega liidetavad

  $O(\ 100\ n^3 \log n + 28n^3 + 34\ n + 1000000\ ) = O(n^3 \log n\ )$

- Keerukusklasside sisalduvus

  $O(\log n) \subset O(n) \subset O(n^2) \subset O(n^k) \subset O(k^n) \subset O(n!) \subset O(n^n)$

Mõned märkused

- Kuigi on korrektne öelda, et “$7n - 3$ on $O(n^3)$”, on siiski parem väide “$7n - 3$ on $O(n)$”. Keerukusklassi tuleks väljendada nii täpselt kui võimalik
- Tihti kasutatakse $O(n)$ ja mõeldakse $\Theta(n)$.
- Kasutataks erinevaid tähistusi. Järgnevad tähendava sama:

  $7n - 3$ on $O(n^3)$ $\quad$ $7n - 3 \in O(n^3)$ $\quad$ $7n - 3 = O(n^3)$

- Spetsiaalsed keerukusklassid:

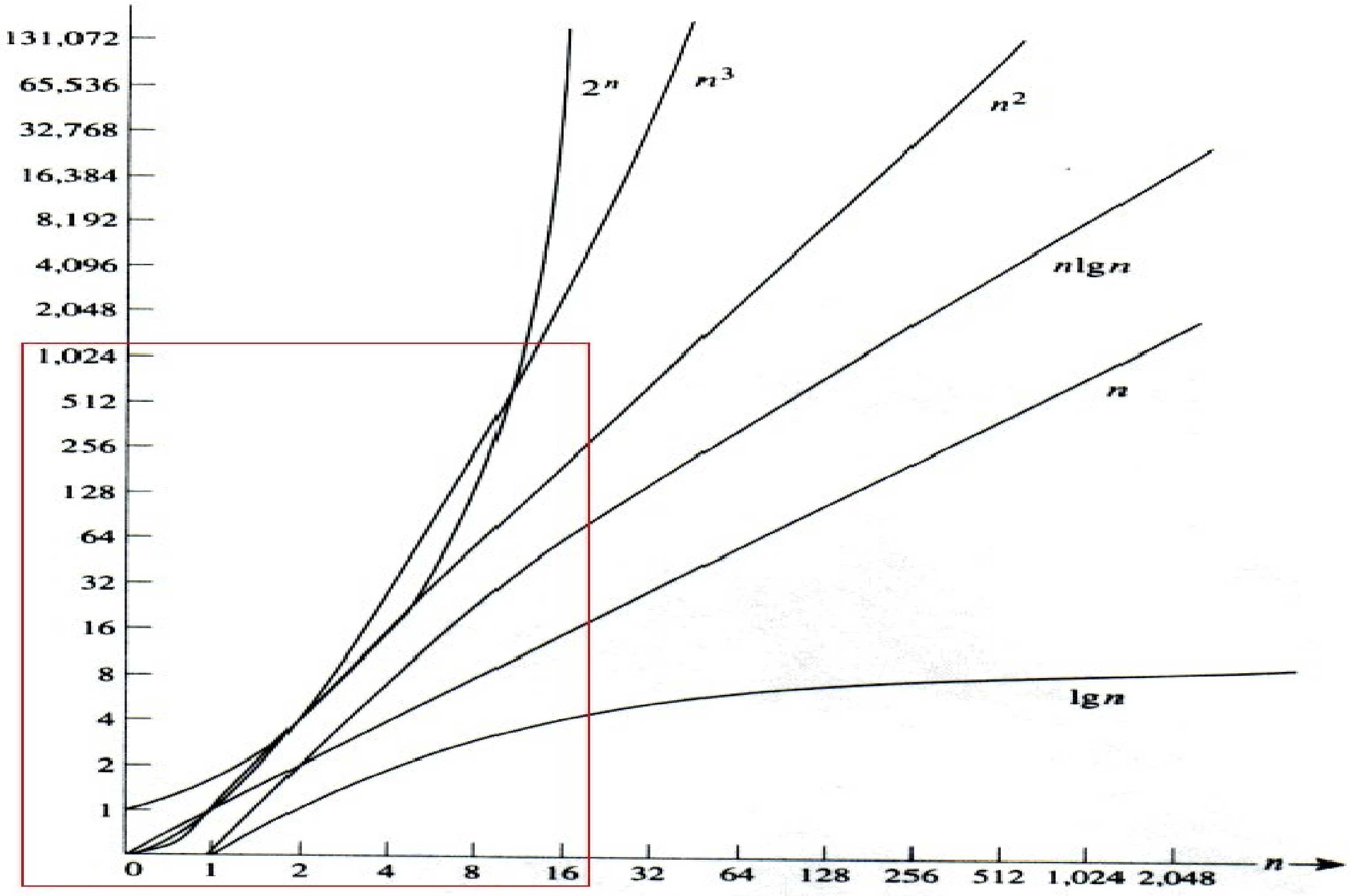
  logaritmiline: **$O(\log n)$**

  lineaarne: **$O(n)$**

  polünomiaalne: **$O(n^k)$, $k \geq 1$**

  eksponentsiaalne: **$O(a^n)$, $n > 1$**

Erinevad keerukusklassid



Polünomiaalsed ja raskeltarvutatavad algoritmid

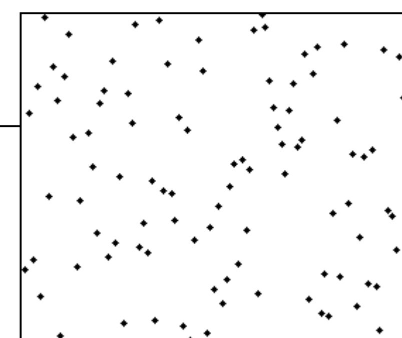
- Polünomiaalne ajaline keerukus
  - Algoritm on polünomiaalne kui ta on $O(n^d)$ mingi täisarvu d korral
  - Polünomiaalseid algoritme peetakse efektiivseteks
    - Nad lahendavad ülesande tavaliselt mõistliku ajaga!

- Raskeltarvutatavad probleemid
  - probleemid, millel pole teada polünomiaalset algoritmi
  - *tuleme selle tähtsa ülesannete klassi juurde hiljem kursuse jooksul*

Mullsorteerimine (*Bubble/exchange sort*)

- Valime põhioperatsiooniks
  `if (a[j+1] < a[j])`
- Vältimatu keerukus
  $T(n) = n(n-1)/2 \in O(n^2)$
- Kui on olemas vältimatu keerukus, siis on halvima, parima ja keskmise juhu keerukus sellega võrdne
- Mis on keerukuseks kui valida põhioperatsiooniks omistamine?
- See sorteerimisalgoritm on lihtne, aga ebaefektiivne suurte andmehulkade juures
- Võib osutuda parimaks väikeste andmehulkade korral

```
for (i=0; i<n-1; i++)
{
  for (j=0; j<n-1-i; j++)
    /*compare neighbors*/
    if (a[j+1] < a[j])
    {
      /*swap a[j]. a[j+1]*/
      tmp = a[j];
      a[j] = a[j+1];
      a[j+1] = tmp;
    }
}
```



Mõned vajalikud terminid

- $\lfloor x \rfloor$ floor of *x* (ümardamine alla)

  $\lfloor 7/2 \rfloor = 3$
- $\lceil x \rceil$ ceiling of *x* (ümardamine üles)

  $\lceil 7/2 \rceil = 4$
- $\sum$ summa

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$$

Logaritmid

- Kümnendlogaritm numbrist *x* on aste, millesse tuleb tõsta arv 10, et saada *x*.
- Logaritm alusel *b* numbrist *x* on aste, millesse tuleb tõsta arv *b*, et saada *x*

$\log 10 = 1$	$10^{1} = 10$	
$\log 1000 = 3$	$10^{3} = 1000$	
$\log 0.01 = -2$	$10^{-2} = 0.01$	
$\log 1 = 0$	$10^{0} = 1$	
$\log_2 8 = 3$	$2^{3} = 8$	
$\log_2 7 \approx 2.807$	$2^{2.807} \approx 7$	
$\lg x \equiv \log_2 x$		kahendlogaritm
$\ln x \equiv \log_e x$	$(e \approx 2.7182818)$	naturaallogaritm

Logaritmide omadusi

- $\log 1 = 0$
- $a^{\log_a x} = x$
- $\log(xy) = \log x + \log y$
- $\log x/y = \log x - \log y$
- $\log x^y = y \log x$
- $y^{\log x} = x^{\log y}$
- $\log_a x = \log_b x / \log_b a$

Põhilised terminid sellest loengust

- Keerukus - põhioperatsioonide arvu sõltuvus sisendi suurusest
- Keerukuskriteeriumid - **halvima**, parima, keskmise juhu keerukus
- Asümptootiline keerukus - keerukus sisendi piiramatul kasvamisel
- Keerukuse rajad - ülemine (O), alumine (Ω) ja täpne (Θ)
- Keerukusklassid -
  - konstantne O(1)
  - logaritmiline O(log n)
  - lineaarne O(n)
  - polünomiaalne O(n^k)
  - eksponentsiaalne O(k^n)