

## 1.4 Funktioiden kertaluokat

- $f$  on kertaluokkaa  $O(g)$ , merk.  $f = O(g)$ , jos joillain  $c > 0$ ,  $m \in \mathbf{N}$  pätee

$$f(n) \leq cg(n) \quad \text{aina kun } n \geq m$$

- $f$  on samaa kertaluokkaa kuin  $g$ , merk.  $f = \Theta(g)$ , jos joillain  $a, b > 0$ ,  $m \in \mathbf{N}$  pätee

$$ag(n) \leq f(n) \leq bg(n) \quad \text{aina kun } n \geq m$$

- $f$  on alempaa kertaluokkaa kuin  $g$ , merk.  $f = o(g)$ , jos kaikilla  $c > 0$  on olemassa  $m \in \mathbf{N}$  jolle

$$f(n) < cg(n) \quad \text{aina kun } n \geq m$$

- $g$  on alaraja funktiolle  $f$ , merk.  $f = \Omega(g)$ , jos jollain  $c > 0$  pätee

$$f(n) \geq cg(n) \quad \text{äärettömän monella } n$$

- $f$  ja  $g$  ovat asymptoottisesti samat, merk.  $f \sim g$ , jos

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1.$$

Usein käytetään hieman epätäsmällisiä merkintöjä tyyliin " $f(n) = O(g(n))$ ", esim. " $n^2 = O(n^3)$ "

## Esimerkkejä

$$\begin{aligned}2n^4 + 6n^8 &= \Theta(n^8) \\ n \log n &= o(n^{1+\epsilon}) \text{ kaikilla } \epsilon > 0 \\ n \log n &= \Omega(n) \\ \log_a n &= \Theta(\log_b n) \text{ kaikilla } a, b > 1 \\ 5n^2 + 10n \log n + 7n &\sim 5n^2\end{aligned}$$

Potenssi vastaan eksponenttifunktio: kaikilla  $\alpha > 0$  ja  $\beta > 1$  pätee

$$n^\alpha = o(\beta^n).$$

Logaritmi vastaan potenssi: kaikilla  $\alpha, \beta > 0$  pätee

$$(\log n)^\alpha = o(n^\beta).$$

Nämä seuraavat suoraan tuloksista (esim. Diff. int. I)

$$\begin{aligned}\lim_{n \rightarrow \infty} \frac{\log n}{n^\alpha} &= 0 \text{ kaikilla } \alpha > 0 \\ \lim_{n \rightarrow \infty} \frac{n}{\beta^n} &= 0 \text{ kaikilla } \beta > 1\end{aligned}$$

Muita hyödyllisiä logaritmin ominaisuuksia:

$$\begin{aligned}a^{\log_a x} &= x \\ \log_a(xy) &= \log_a x + \log_a y \\ \log_a(x^y) &= y \log_a x \\ \log_a x &= \frac{\log b}{\log a} \log_a x\end{aligned}$$

Jatkossa merkitään  $\log$  kun kantaluku on 2 tai kantaluvulla ei ole väliä. Luonnollista logaritmia merkitään  $\ln$ .

## Kertaluokkien yleisiä ominaisuuksia

Jos  $f = O(g)$  ja  $g = O(h)$  niin  $f = O(h)$ .

Kaikilla  $c > 0$  pätee  $f(n) = O(g(n))$  jos ja vain jos  $f(n) = O(cg(n))$ .

Aina  $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$ .

**Lause** Olkoot  $f, g: \mathbf{N} \rightarrow \mathbf{R}^+$ .

- Jos

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

niin  $f(n) = o(g(n))$ .

- Jos jollain  $0 < c < \infty$  pätee

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

niin  $f(n) = \Theta(g(n))$ .

- Jos

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

niin  $f(n) = \Omega(g(n))$ .

Algoritmilla on **polynominen aikavaativuus** jos sen aikavaativuus  $T(n) = O(n^k)$  jollain vakiolla  $k$ .  
(Vastaavasti tilavaativuus jne.)

**Esimerkkejä** Seuraavassa  $n$  on syötteen koko,  $T(n)$  aikavaativuus ilmoitettuna tiettyjen alkeisoperaatioiden lukumääränä, ja koneen oletetaan suorittavan  $10^6$  alkeisoperaatiota sekunnissa. Laskenta-aika on taulukossa.

$T(n)$	$n$				pol.?
	20	50	100	200	
$1000n \log n$	0,1 s	0,3 s	0,6 s	1,5 s	on
$10n^3n$	0,02 s	1 s	10 s	1 min	on
$n^{\log n}$	0,4 s	1,1 h	220 vrk	12000 v	ei
$2^n$	1 s	35 v			ei

Sama tilanne mutta taulukoidaan suurin mahdollinen käsiteltävä syöte kun aikaraja on annettu.

$T(n)$	1 s	$10^4$ s (= 2,7 h)	$10^8$ s (= 3 v)
$1000n \log n$	140	520000	$3 \cdot 10^9$
$10n^3n$	46	1000	21000
$n^{\log n}$	22	54	112
$2^n$	19	26	46

Tarkastellaan samaa asiaa graafisesti. Ohessa arvoilla  $n = 1, \dots, 120$  kuvaajat funktioille

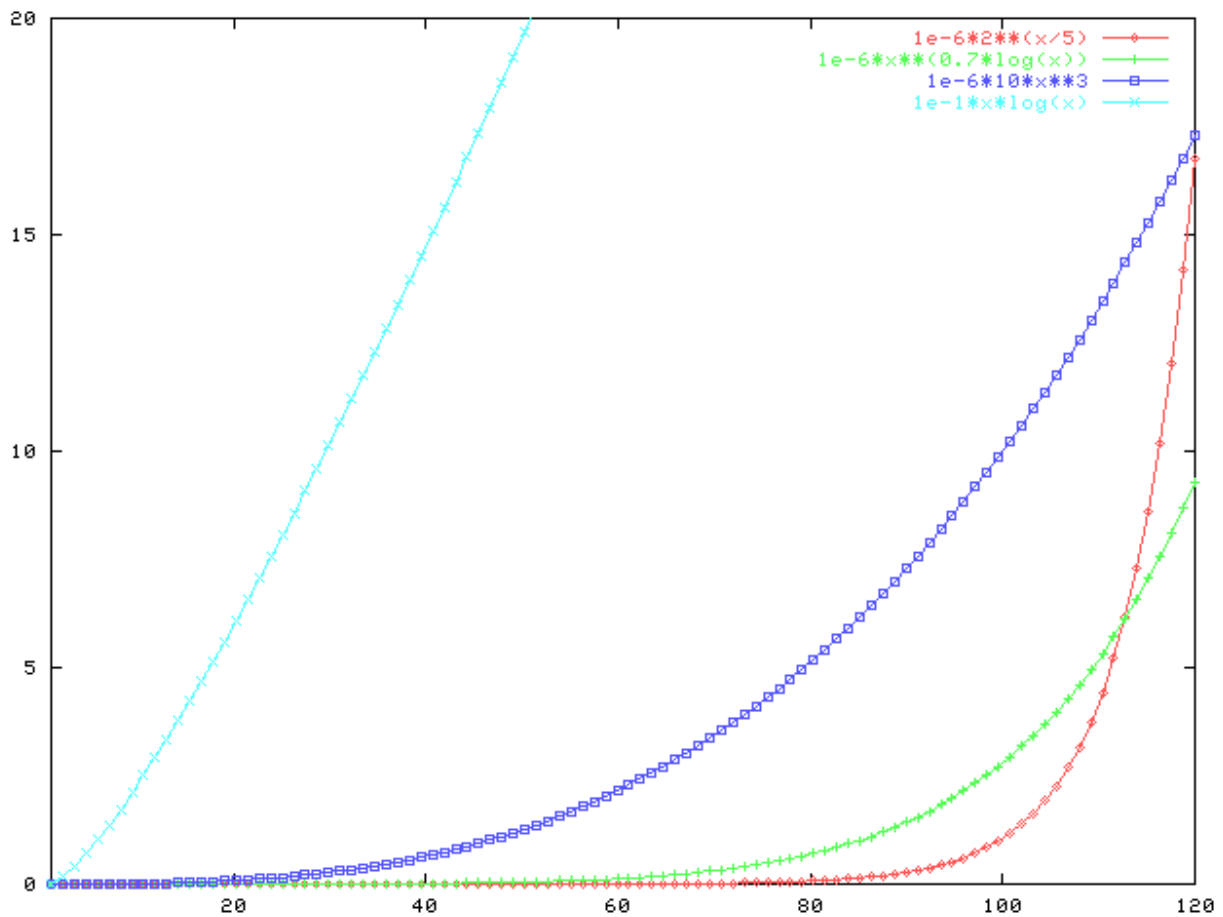
$$10^{-6} \cdot 2^{n/5}$$

$$10^{-6} \cdot n^{0,7 \cdot \ln n}$$

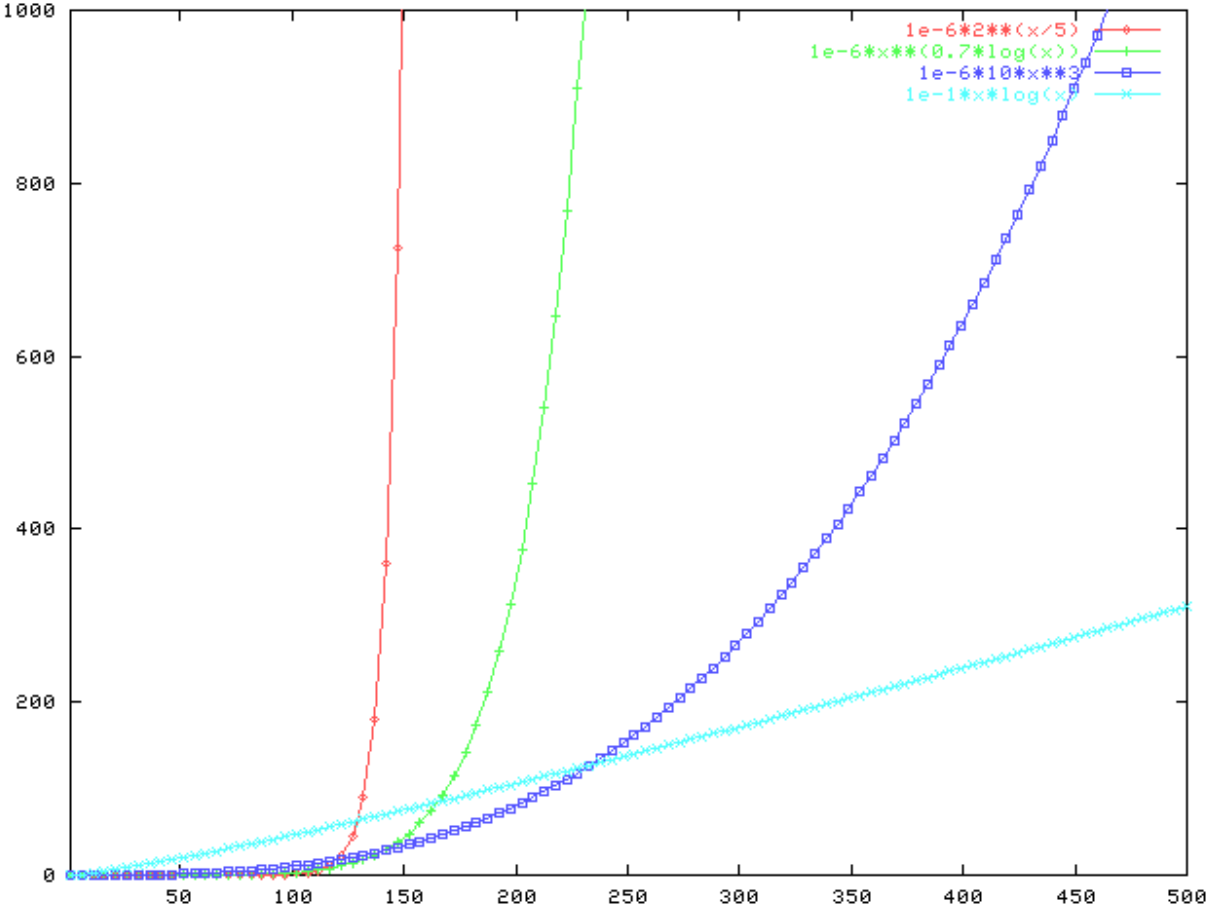
$$10^{-5} \cdot n^3$$

$$0,1 \cdot n \ln n$$

(Vakiokertoimet valittu siten että funktiot suunnilleen sopivat samaan kuvaan.)



Kun jatketaan arvoihin  $n = 100, \dots, 500$ , nähdään miten ei-polynomiset funktiot räjähtävät käsistä.



## Miksi polynomisuus on tärkeää

- *polynominen algoritmi*: laskenta-ajan (tai -tehon) kasvattaminen vakiokertoimella kasvattaa myös mahdollisten ongelmien kokoa *vakiokertoimella eksponentiaalinen algoritmi*: ongelmien koko kasvaa vain *vakiotermillä*
- Polynomiset algoritmit käyvät asteittain hitaammiksi ja hitaammiksi. Eksponentiaaliset algoritmit törmäävät jossain vaiheessa kuin seinään, minkä jälkeen laskentatehon lisäys ei juuri hyödytä. (Poikkeus: korkean kertaluvun polynomit; näitä ei kuitenkaan käytännössä juuri esiinny.)
- teoreettiset ominaisuudet: "polynominen aikavaativuus" sama kaikissa laskennan malleissa
- sulkeumaominaisuudet: kun aikavaativuudeltaan polynomisia algoritmeja yhdistellään esim. **for**-lauseisiin, aikavaativuus säilyy polynomisena.

## Kuitenkin otettava huomioon

- pahin tapaus ei välttämättä tyypillinen
- toisinaan tarkastellaan pieniä syötteitä
- kertakäyttösovellukset: koodauksen helppous
- käytännössä suoritusaikaa voi dominoida I/O, haut hitaasta muistista tms.
- numeeriset algoritmit: stabiilius, aritmetiikan tarkkuus

## 2. Algoritmien analyysitekniikoita

Kerrataan kurssilta Tietorakenteet tutut perustekniikat ja opitaan uusia erityisesti rekursiivisten algoritmien ja keskimääräisen analyysin tarpeisiin.

Tämän luvun jälkeen opiskelija

- osaa kuvata rekursiivisen algoritmin aika- ja tilavaativuuden rekursioyhtälöllä
- osaa soveltaa tärkeimpiä rekursioyhtälöiden ratkaisumenetelmiä, etenkin generoivia funktioita ja ns. master-teoreemaa
- osaa mallintaa satunnaisilmiötä Markovin ketjulla ja tarkastella sen tasapainojakaumaa
- osaa analysoida tasoitettua aikavaatimusta potentiaalimenetelmän avulla

## 2.1 Iteratiivisen algoritmin aikavaativuus

Tämä on tuttua kurssilta Tietorakenteet. Käydään kertausesimerkkinä läpi lisäysjärjestämisalgoritmi.

```
insert-sort( $A[1 \dots n]$ ):  
1.   for  $j := 2$  to  $n$  do  
2.      $x := A[j]$   
3.      $i := j - 1$   
4.     while  $i > 0$  and  $A[i] > x$  do  
5.        $A[i + 1] := A[i]$   
6.        $i := i - 1$   
7.     end while  
8.      $A[i + 1] := x$   
9.   end for
```

Millä tahansa  $n$ ,  $i$  ja  $j$  rivit 5 ja 6 vievät vakioajan. Merkitään tätä

$$T_{5..6}(n, i, j) = c_1.$$

**while**-silmukka kullakin  $j$  korkeintaan  $j$  kertaa:

$$T_{4..6}(n, j) \leq c_2 + (j - 1)(c_1 + c_3)$$

missä  $c_1$  tulee siis riveistä 5 ja 6,  $c_3$  vastaa ehtotestin suoritusta ja  $c_2$  silmukan alustusta.

Samoin rivit 2, 3 ja 7 menevät vakioajassa joten

$$T_{2\dots 7}(n, j) \leq c_2 + c_4 + (j - 1)(c_1 + c_3).$$

Lopulta (kun  $c_5$  esittää **for**-silmukan alustamista) saadaan

$$\begin{aligned} T_{1\dots 7}(n) &= c_5 + \sum_{j=2}^n T_{2\dots 7}(n, j) \\ &\leq c_5 + (n - 1)(c_2 + c_4) + (c_1 + c_3) \sum_{j=2}^n (j - 1) \\ &= c_5 + (n - 1)(c_2 + c_4) + (c_1 + c_3) \frac{1}{2} n(n - 1) \end{aligned}$$

joten

$$T(n) = O(n^2).$$

## Yleisiä periaatteita

- $T("x := e"), T(" \text{read } x"), T(" \text{write } e")$  vakioita (poikkeuksia: funktiokutsut, tarkka aritmetiikka, taulukon indeksointi)
- $T("P_1; P_2; \dots; P_k") = T(P_1) + T(P_2) + \dots + T(P_k) = \Theta(\max \{ T(P_1), \dots, T(P_k) \})$
- $T(" \text{if } e \text{ then } P_1 \text{ else } P_2") = \Theta(\max \{ T(e), T(P_1), T(P_2) \})$
- $T(" \text{while } e \text{ do } P") = (\text{suorituskertojen lkm}) \cdot \Theta(1 + T(e) + T(P))$

## Hyödyllisiä summakaavoja

$$\sum_{i=0}^n i = \frac{n(n+1)}{2} = \Theta(n^2) \quad (1)$$

$$\sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3) \quad (2)$$

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1 \quad (3)$$

$$\sum_{i=0}^n a^i = \frac{a^{n+1} - 1}{a - 1} \quad \text{kun } a \neq 1 \quad (4)$$

$$\sum_{i=0}^n ia^i = \frac{na^{n+2} - (n+1)a^{n+1} + a}{(a-1)^2} \quad \text{kun } a \neq 1 \quad (5)$$

$$\sum_{i=0}^{\infty} a^i = \frac{1}{1-a} \quad \text{kun } |a| < 1 \quad (6)$$

$$\sum_{i=0}^{\infty} ia^i = \frac{a}{(1-a)^2} \quad \text{kun } |a| < 1 \quad (7)$$

$$\sum_{i=1}^n \frac{1}{i} = \ln n + \gamma + O(1/n) \quad (8)$$

missä  $\gamma \approx 0,577$  (Eulerin vakio)

Näistä on syytä muistaa (tai osata johtaa) ainakin (1), (4) ja (6) tarkalleen ja loput kertaluokaltaan.

## 2.2 Rekursioyhtälöt

Tunnetaan myös nimillä differenssiyhtälöt ja palautuskaavat; keskeinen väline rekursiivisten algoritmien analysoimisessa

**Motivoiva esimerkki:** lomitusjärjestäminen  
Olkoon  $T(n)$  seuraavan proseduurin mergesort( $A, p, q$ ) aikavaatimus kun  $q - p + 1 = n$ .

Tässä  $A$  on järjestettävä taulukko ja merge( $A, p, r, q$ ) lomittaa osataulukot  $A[p \dots r]$  ja  $A[r + 1 \dots q]$  lineaarisessa ajassa.

```
merge-sort( $A, p, q$ ):  
1.   if  $p < q$  then                                 $\Theta(1)$   
2.        $r := \lfloor (p + q)/2 \rfloor$                         $\Theta(1)$   
3.       merge-sort( $A, p, r$ )                           $\Theta(T(\lfloor n/2 \rfloor))$   
4.       merge-sort( $A, r + 1, q$ )                       $\Theta(T(\lceil n/2 \rceil))$   
5.       merge( $A, p, r, q$ )                             $\Theta(n)$   
   end if
```

Saadaan siis joillain vakioilla  $c_1$ ,  $c_2$  ja  $c_3$

$$\begin{aligned} T(1) &= c_1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + c_2 n + c_3 \quad (9) \\ &\quad \text{kun } n \geq 2 \end{aligned}$$

Yksinkertaistetaan yhtälöä jättämällä pois alemman kertaluvun termit ja olettamalla että pyöristyksiä ei tarvita:

$$\begin{aligned} T(1) &= c_1 \\ T(n) &= 2T(n/2) + c_2n \quad \text{kun } n = 2^k, k \geq 1 \quad (10) \end{aligned}$$

Ratkaistaan ensin tämä yksinkertaisempi yhtälö (10) ja osoitetaan sitten, että sama ratkaisu toteuttaa myös alkuperäisen yhtälön (9).

Ratkaistaan (10) **arvausta sovittamalla**:

1. *arvataan* ratkaisun kertaluokka  $T(n) = \Theta(n \log n)$
2. *sovitetaan* vakiokertoimet niin että yhtälö toteutuu

**Lause** Olkoon  $T$  yhtälön (10) ratkaisu, kun rajoitutaan muotoa  $n = 2^k$  oleviin argumentteihin. Nyt

$$T(n) = \Theta(n \log n).$$

**Todistus** Osoitetaan ensin yläraja  $T(n) = O(n \log n)$ . Tämän kanssa on yhtäpitävää että joillain  $a, b > 0$  pätee

$$T(n) \leq an \log n + b \quad (11)$$

kaikilla  $n$ .

Todistus tapahtuu induktiolla. Todetaan ensin, että (11) pätee arvolla  $n = 1$  jos valitaan  $b \geq c_1$ .

Oletetaan nyt, että jollain  $m \geq 2$  yhtälö (11) pätee kun  $n < m$ . Soveltamalla yhtälöä (10) ja induktio-oletusta saadaan nyt

$$\begin{aligned} T(m) &= 2T(m/2) + c_2m \\ &\leq 2\left(a\frac{m}{2} \log \frac{m}{2} + b\right) + c_2m \\ &= am \log m + m(c_2 + a \log(1/2)) + 2b \\ &= am \log m + b + (b + m(c_2 - a)) \\ &\leq am \log m + b \end{aligned}$$

jos lisäksi  $b + m(c_2 - a) \leq 0$ . Koska  $m \geq 2$ , kumpikin ehto toteutuu esim. valinnalla  $b = c_1$  ja  $a = c_2 + c_1/2$ .

Tällä valinnalla siis myös (11) toteutuu, eli

$$T(n) \leq (c_2 + c_1/2)n \log n + c_1$$

kaikilla  $n$ .

Lauseen toinen puoli eli alaraja  $T(n) = \Omega(n \log n)$  todistetaan suoraviivaisella induktiolla.

**Väite:**  $T(n) \geq c_2 n \log n$  kaikilla  $n$

**Perustapaus:**  $T(1) = c_1 \geq 0 = c_2 \cdot 1 \cdot \log 1$

**Induktioaskel:** Induktio-oletuksesta saadaan

$$\begin{aligned} T(n) &= 2T(n/2) + c_2 n \\ &\geq 2c_2 \frac{n}{2} \log \frac{n}{2} + c_2 n \\ &= c_2 n \log n + c_2 n \log(1/2) + c_2 n \\ &= c_2 n \log n. \end{aligned}$$

Siis  $T(n) = \Theta(n \log n)$ .



Jotta nyt saataisiin lomitusjärjestämisen aikavaativuudeksi  $\Theta(n \log n)$ , pitää vielä osoittaa, että yhtälöiden (9) ja (10) ratkaisut ovat samaa kertaluokkaa, ts. että

1. alemmanasteinen termi  $c_3$  ei vaikuta tulokseen ja
2. tulos pätee myös kun  $n$  ei ole kakkosen potenssi.

Olkoon  $T$  alkuperäisen yhtälön (9) ratkaisu.

Kohtaa (i) varten määritellään kaikilla  $n = 2^k$

$$\begin{aligned} T_1(1) &= c_1 \\ T_1(n) &= 2T(n/2) + c_2n \quad \text{kun } n = 2^k > 1 \text{ ja} \end{aligned}$$

$$\begin{aligned} T_2(1) &= c_1 \\ T_2(n) &= 2T(n/2) + (c_2 + c_3)n \quad \text{kun } n = 2^k > 1 \end{aligned}$$

Koska edellisen lauseen nojalla sekä  $T_1(n)$  että  $T_2(n)$  ovat  $\Theta(n \log n)$  ja selvästi

$$T_1(n) \leq T(n) \leq T_2(n),$$

nähdään että  $T(n) = \Theta(n \log n)$  kun  $n = 2^k$ .

Siis on olemassa  $a, b > 0$  ja  $n_0 > 0$  joilla

$$an \log n \leq T(n) \leq bn \log n$$

kun  $n \geq n_0$  ja  $n = 2^k$  jollain  $k$ .

Olkoon nyt  $n \geq n_0$  mielivaltainen ja  $k$  sellainen, että  $2^k \leq n < 2^{k+1}$ . Koska selvästi  $T$  on kasvava, saadaan

$$a2^k k \leq T(2^k) \leq T(n)$$

ja

$$T(n) < T(2^{k+1}) \leq b2^{k+1}(k+1).$$

Koska  $\log n - 1 < k \leq \log n$ , saadaan

$$\frac{a}{2}n(\log n - 1) < T(n) \leq 2bn(\log n + 1)$$

mistä seuraa  $T(n) = \Theta(n \log n)$ .

Yllä on esimerkin vuoksi tarkasti näytetty miten yksinkertaistetun yhtälön (10) ratkaisusta päästään alkuperäisen yhtälön (9) ratkaisuun.

Jatkossa ei enää kiinnitetä huomiota tähän suoraviivaiseen mutta kiusalliseen välivaiheeseen.

Rajoitumme oleellisempaan ongelmaan eli tyyppiä (10) olevien yhtälöiden ratkaisemiseen.