Algoritmide keerukus

Algoritmi ja keerukuse mõiste. Hindamine. Keerukusklassid.

Algoritm

Algoritm on lõplik käskude (ingl *instruction*), sammude hulk, mis annab täpse operatsioonide järjestuse mingi probleemi (või probleemide klassi) lahendamiseks.

Algoritm on ülesande lahendamiseks täpselt määratletud reeglite lõplik korrastatud kogum (Infotehnoloogia sõnastik EVS-ISO/IEC 2382-1:1998)

Igal käsul on täpne tähendus ja lõplik täitmise aeg.

Arvutiteaduses on **algoritm** meetod probleemi lahendamiseks *(ingl problem solving)*, mida saab realiseerida arvutiprogrammi abil.

Algoritm on probleemi lahendamise plaan (ingl *blueprint*). Kui algoritm olemas, võib selle tõlkida programmeerimiskeelde.

Algoritmi omadused

Algoritm peab olema **lihtne** ja **lühike**.

Ta peab olema **tõhus** (soovitud häid ja vajalikke tulemusi andev). Soovitud tulemus peab tekkima lõpliku ajaga.

Ta peab olema **üheselt kirjeldatud**, ei tohi olla mitmeti mõistetav ning loogika peab olema **selge**.

Ta peab olema **lõplik**, st algoritmi töö lõppeb peale lõpliku arvu sammude läbimist. Sammude arv võib olla väga suur.

Algoritmi omadused

Sisend: algoritm aktsepteerib sisendandmeid tema jaoks kirjeldatud hulgast.

Väljund: algoritm tekitab tulemused, millel on täpselt määratud seos sisendandmetega.

Üldkehtivus: algoritm peab sobima kõigi andmete (isendite) jaoks kirjeldatud sisendite hulgast.

Õigsus ehk veatus

Algoritm peab olema **õige** ehk **veatu** (ingl *correct*) – **iga** lubatud sisendi korral lõpetab ta töö õige väljundiga.

Veatu algoritm lahendab **arvutusprobleemi** (ingl *computational problem*).

Vigane algoritm võib osa sisendite korral saada õige tulemuse, mõne sisendi korral tööd mitte lõpetada ja osa sisendite korral anda vale vastuse.

Tõhusus, analüüsimine

Algoritm peab olema **tõhus** ehk **efektiivne** (ingl *efficient*)**.** Efektiivsuse peamiseks mõõdupuuks on algoritmi tööaeg.

Ühte probleemi saab lahendada erinevate algoritmidega. Erinev tõhusus mõjutab tihti tööaega rohkem kui arvuti tehnilised näitajad.

Sobivama (efektiivsema) algoritmi leidmiseks on vaja algoritmi kandidaate analüüsida (ingl *analyzing an algorithm*).

Algoritmi analüüsimiseks nimetatakse tema ressursivajaduse ennustamist. Ressursiks on mälu ja aeg.

Tööaja hindamine

Kuidas hinnata algoritmi täitmiseks kuluvat aega? Kuidas seda mõõta, et algoritme võrrelda?

- sekundites (paneme käima ja mõõdame; erineva kiirusega arvutid)
- koodiridades (erinevad stiilid ridade kirjutamisel)
- elementaartehetes (erinevad tehted - erinev aeg)

Parem on algoritmi analüüsida sõltumatult keelest, riistvarast, op-süsteemist.

Võrreldav on, kuidas sõltub algoritmi tööaeg sisendi suurusest. Kuid selle jaoks on vaja määrata lihtoperatsioonide arvu seos sisendi suurusega.

Sisendi suurus -> aeg

Ajakulu saab hinnata lihtoperatsioonide arvuga. Operatsiooni piiritlemine on keeruline, arvestada tuleb arvuti eripäradega.

Üks võrdlemine (omistamine, tehe, ...): üks operatsioon: lihtsustatud korras – üks lause: üks operatsioon.

Kasutatakse hüpoteetilise arvuti mõistet, kus kõik lihtoperatsioonid loetakse võrdseteks sammudeks.

Mida rohkem andmeid, seda rohkem operatsioone kulub nende töötlemiseks.

Näide: otsimine

```
Korda kõigi andmetega
  Kui andmeühik == otsitav
    Teata "Leitud"
    Lõpeta otsimine
```

Millest sõltub näites operatsioonide arv?

Sisendi iseloom -> aeg

- Kui otsitav on esimene?
- Kui otsitav on keskel?
- Kui otsitav on viimane?
- Kas saab olla veel kehvemat varianti?

Algoritmi tööaeg ja keerukus

Algoritmi tööaeg (ingl *running time*) konkreetse sisendi suuruse korral on lihtoperatsioonide arv.

Sarnaselt defineeritakse ka **algoritmi keerukus** (ingl *complexity*)

Võib rääkida algoritmi tööajast parimal, halvimal ja keskmisel juhul.

Määrata tuleb enamasti **algoritmi tööaeg halvimal juhul** (ingl *worst-case running time*), sest nii saame tööaja ülemise piiri.

Tööaja arvutus on suur üldistus.

Algoritmi keerukus

Analüüsides **algoritmi keerukust** saab anda hinnagu tema tööajale.

Algoritmi keerukus:

- halvimal juhul *(ingl worst-case complexity);*
- parimal juhul *(ingl best-case complexity);*
- keskmisel juhul *(ingl average-case complexity).*

Kriitilistel juhtudel on kindlasti oluline keerukus halvimal juhul.

Algoritmi keerukus kui funktsioon

Algoritmi keerukus (ingl *complexity*) on funktsioon, mis seab andmete mahule vastavusse algoritmi sammude arvu.

- Sammu võib tõlgendada erinevalt – see võib olla üks tehe, üks tsüklitingimus või ka üks lause

Algoritmi **keerukusfunktsiooni kasvukiirus** näitab, kui kiiresti kasvab vastava programmi ressursivajadus töödeldavate andmete mahu kasvades.

Algoritmi keerukusfunktsiooni uurimine lubab anda hinnangu algoritmi efektiivsusele.

Algoritmi keerukusfunktsioon

Algoritmi keerukusfunktsiooni leidmiseks on võimalik kokku arvutada kõik sammud, mida arvuti teeb ülesande lahendamiseks.

Seda ei ole võimalik väljendada konkreetse arvuna, vaid andmete hulgast (n) sõltuva valemina.

Hüpoteetiliselt on ühe operatsiooni kestvus üks samm, siis n elemendi töölemisel kuluks aega n sammu. Sellele võivad näiteks lisanduda aega C_1 võtvad operatsioonid, mis ei ole aga iga andmega seotud: $n + C_0$

Asümptootiline keerukus

Liiga täpsete hinnangutega (sammude arvude lugemisega) ei ole suuremat pihta hakata. Seda enam, et iga sammu täpne täitmise aeg ei ole teada.

Analüüsi võib lihtsustada ja saada hinnang, mis annab piisava üldise iseloomustuse.

Hinnangud antakse suurte (piiramatult kasvavate) sisendite jaoks.

Funktsiooni `f(n)` **asümptootiline käitumine** kirjeldab `f(n)` kasvu, kui andmete maht `n` piiramatult kasvab.

Asümptootilised hinnangud

Matemaatiliselt võib kohata tähistusi (c, c_1, c_2 on konstandid)

$f(n) = \Theta(g(n))$ – (kreeka täht theeta) asümptootiliselt täpne hinnang ehk $f(n)$ kasv jääb $c_1 * g(n)$ ja $c_2 * g(n)$ vahele

$f(n) = O(g(n))$ – ülevalt piiratud hinnang ehk $f(n)$ ei kasva kiiremini kui $c * g(n)$

Viimast kasutatakse algoritmide hindamisel.

Algoritmi asümptootilise keerukuse hindamisel uuritakse, kuidas hakkab ta käituma **väga suurte andmemahtude korral**, alates mingist n_0-st.

Asümptootilised hinnangud

S. Skiena, Algorithm Design Manual

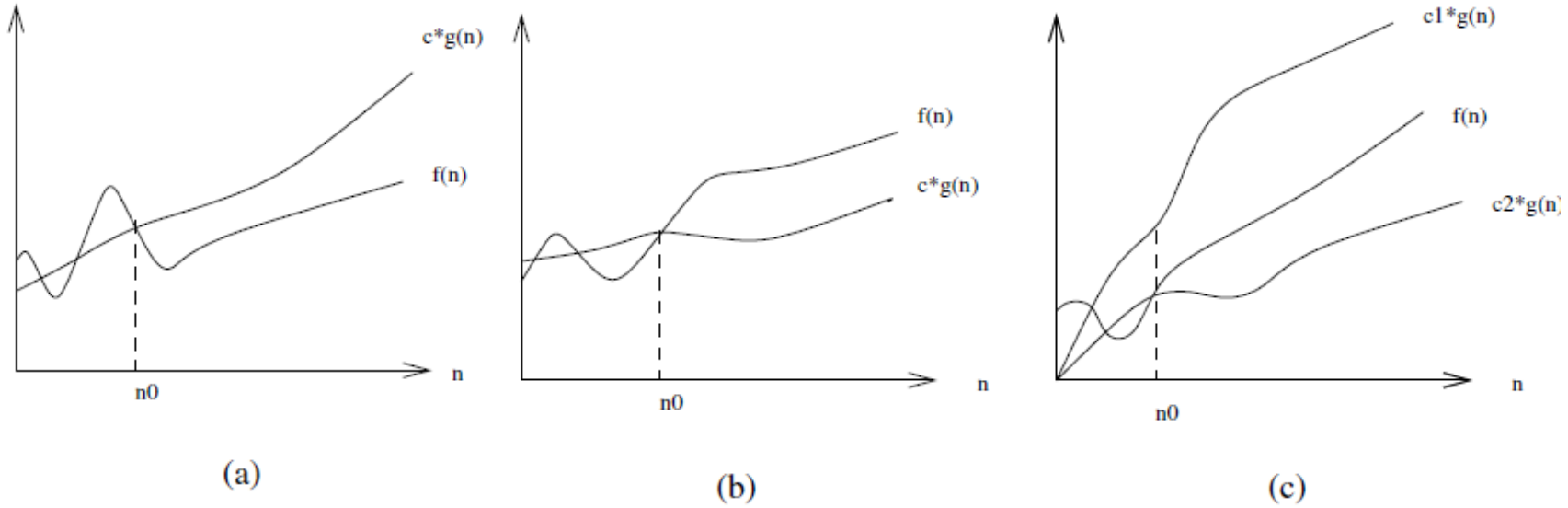

Figure 2.3: Illustrating the big (a) O, (b) Ω, and (c) Θ notations

Suure O tähistus

Suure O tähistust *(ingl Big Oh notation)* kasutatakse algoritmide keerukusklasside tähistamiseks.

`O(g(n))` on funktsioonide hulk, mis ei kasva kiiremini kui `g(n)`.

`g(n)` omakorda on funktsioon, mis kirjeldab algoritmi sammude arvu seost sisendi mahuga `(n)`.

- Nt võib funktsiooniks `g(n)` olla `n`, n^2 jms

Keerukusklassid

Algoritmide süstematiseerimiseks on kasutusel O-tähistusele tuginevad **keerukusklassid:**

- O(1) – konstantne keerukus
- O(log n) – logaritmiline keerukus
- O(n) – lineaarne keerukus
- O(n log n) – linearitmiline keerukus
- $O(n^2)$ – ruutkeerukus
- $O(n^3)$ – kuupkeerukus, üldisemalt polünomiaalne k
- $O(2^n)$ – eksponentsiaalne keerukus

Keerukusklassi määramine

Algoritmi keerukusklassi määramiseks tuleb

- otsustada, millised operatsioonid kokku lugeda: nt lugeda sammuks lause;
- väljendada sammude arv sisendi suurust (n) arvestades: nt korrutades tsüklis tehtavad tegevused n-ga;
- lihtsustada tulemus - polünoomist jääb kõrgeim aste, kaovad konstantsed kordajad jms:
  - Näiteks: $3n^2 + 5n + 2$ lihtsustub klassiks $O(n^2)$

Keerukusklassid

Tuntud algoritmide jaoks on keerukusklassid määratud ja neid tuleks usaldada.

Lihtsamatel juhudel saab ka ise koodi vaadates anda hinnagut keerukusklassile.

Näiteks:

- 1 tsükkel üle kõigi andmete – $O(n)$,
- 2 tsüklit üksteise sees üle kõigi andmete – $O(n^2)$

Tööaja kasv

Programmi tööaja suhteline kasv andmete mahu kasvades 5 x (5lt 25le).

Prog tööaeg. c f(n)	Suhteline suurenemine f(25) / f(5)
c_1	1
$c_2 \log n$	2
$c_3\ n$	5
$c_4\ n \log n$	10
$c_5\ n^2$	25
$c_6\ n^3$	125
$c_7\ 2^n$	1 048 576

Andmed ja arvuti kiirus

Ajaline keerukus	1 MOPS	10 MOPS	100 MOPS
n	1 miljon	10 miljonit	100 miljonit
n^2	1 000	~ 3 162	~10 000
n^3	100	~ 215	~ 464
2^n	~20	~ 23	~ 26
MOPS – miljonit operatsiooni sek-s		Töödeldavate andmete mahu kasv arvuti kiiruse kasvades 10 x	Töödeldavate andmete mahu kasv arvuti kiiruse kasvades 100 x