

Algoritmer og datastrukturer

Introduktion til kurset

Rolf Fagerberg

Forår 2022

Hvem er vi?

Underviser:

- ▶ Rolf Fagerberg, Institut for Matematik og Datalogi (IMADA)
Forskningsområde: algoritmer og datastrukturer

Deltagere:

- ▶ BA i Datalogi (2. sem.)
- ▶ BA i Software Engineering (4. sem.)
- ▶ BA i Matematik-Økonomi (4. sem.)
- ▶ BA i Anvendt Matematik (4. sem.)
- ▶ BA sidefag i Datalogi (6. sem.)
- ▶ KA profil i Data Science (8. sem.)

Stor diversitet: forskellige semestre i uddannelsen, forskellige mængder af programmering og af matematiske fag på uddannelsen.

Tre kurser i ét!

DM507/DS814:

- ▶ Algoritmer og datastrukturer (8 ETCS)
- ▶ Programmeringsprojekt i tre dele (2 ETCS)
- ▶ Skriftlig MC eksamen (3 timer)

SE4-DMAD:

- ▶ Algoritmer og datastrukturer (8 ETCS)
- ▶ Diskret matematik (2 ETCS)
- ▶ Skriftlig MC eksamen (3:45 timer)

Diskret matematik har separate forelæsninger (Lene Monrad Favrholt) og øvelsestimer. Tirsdage 12-14 i ca. ti uger.

Itslearning: ét kursusrum for DM507/DS814, ét kursusrum for SE4-DMAD.

Kursets format (DM507/DS814)

Forudsætninger:

Programmering i Java eller Python, lidt matematisk modenhed

Format:

Forelæsninger (f-timer).

Opgaveregning (e-timer). Med instruktør.

Arbejde selv og i studiegrupper

Eksamenform:

Skriftlig eksamen (juni), 8 ECTS:

Multiple-choice (med bøger, noter, computer). Karakter efter 7-skala. Mål: check af kendskab til stoffet. [NB: reeksamen er mundtlig.]

Projekt undervejs, 2 ECTS, Java eller Python (frit valg):

I flere dele. Karakter B/IB. Skal ikke (længere) bestå for at gå til skriftlig eksamen. Mål: træne overførsel af stoffet til praksis (programmering).

Kursets format (SE4-DMAD)

Forudsætninger:

Programmering i Java (eller Python), lidt matematisk modenhed

Format:

Forelæsninger (f-timer).

Opgaveregning (e-timer). Med instruktør.

Arbejde selv og i studiegrupper

Eksamenform:

Skriftlig eksamen (juni), 10 ECTS:

Multiple-choice (med bøger, noter, computer). Karakter efter 7-skala. Mål: check af kendskab til stoffet. [NB: reeksamen er mundtlig.]

Materialer i algoritmer og datastrukturer

Lærebog:

Cormen, Leiserson, Rivest, Stein:
Introduction to Algorithms, 3rd edition, 2009.

Andet læremateriale på kursets webside:

- Slides fra forelæsninger
- Links til videoer
- Opgaver til øvelsestimer
- Tidligere eksamenssæt
- Projektet

Stoffet findes i fuld detalje i *fire* udgaver: lærebog, slides, video af forelæser, video af andre. Brug de dele, som virker bedst for dig (gerne flere dele).

Forventet arbejdsindsats (DM507/DS814)

- ▶ Skim stof før forelæsning: 0,5 timer \Leftarrow mindst vigtig
- ▶ Forelæsning: 2 timer
- ▶ Læs stof efter forelæsning: 1,5 timer
- ▶ Opgaveregning (hjemme): 3 timer \Leftarrow mest vigtig
- ▶ Opgaveregning (klasse): 2 timer \Leftarrow mest vigtig

Ovenstående i gennemsnit 1.5 gang per uge over 14 uger. Dertil følgende én gang:

- ▶ Projektet: 15+15+25 timer
- ▶ Eksamenslæsning: 40 timer
- ▶ Spørgetime og eksamen: 6 timer

I alt: $14 \cdot 1.5 \cdot 9 + 55 + 40 + 6 = 290$ timer

10 ECTS = $1/6$ årsværk = $1650/6$ timer = 275 timer

Forventet arbejdsindsats (SE4-DMAD)

- ▶ Skim stof før forelæsning: 0,5 timer \Leftarrow mindst vigtig
- ▶ Forelæsning: 2 timer
- ▶ Læs stof efter forelæsning: 1,5 timer
- ▶ Opgaveregning (hjemme): 3 timer \Leftarrow mest vigtig
- ▶ Opgaveregning (klasse): 2 timer \Leftarrow mest vigtig

Ovenstående i gennemsnit 1.5 per uge over 14 uger (Rolf) og 0.5 gang per uge over 10 uger (Lene). Dertil følgende én gang:

- ▶ Eksamenslæsning: 40 timer
- ▶ Spørgetime og eksamen: 6 timer

I alt: $(14 \cdot 1.5 + 10 \cdot 0.5) \cdot 9 + 40 + 6 = 280$ timer

10 ECTS = $1/6$ årsværk = $1650/6$ timer = 275 timer

Kursets formål og plads i det store billede

Generelt mål i IT: Få computer til at udføre en opgave.

Relaterede spørgsmål:

- ▶ **Hvordan skrives programmer?**
Programmering, programmeringssprog, software engineering.
- ▶ **Hvordan skal programmet løse opgaven?** \Leftarrow DM507
Algoritmer og datastrukturer, databasesystemer, lineær algebra med anvendelser, data mining og machine learning.
- ▶ **(Hvor godt) er det overhovedet muligt at løse opgaven?**
Nedre grænser, kompleksitet, beregnelighed.
- ▶ **Hvordan fungerer maskinen der udfører opgaven?**
Baggrundsviden om computerarkitektur og operativsystemer.

Fokus: *Hvordan* skal programmet løse opgaven?

Algoritme = løsningsmetode.

Tilpas præcist skrevet ned: præcis tekst, pseudo-kode, flow-diagrammer, formler, ...

Datastruktur = data + effektive operationer herpå.

Forskellige datastrukturer gemmer forskellige typer data og/eller tilbyder forskellige operationer. Har stor anvendelse som delement i algoritmer.

Relevante handlinger for en programmeringsopgave:

1. **Find** (mindst) én algoritme som løser opgaven.
2. **Sammenlign** flere algoritmer som løser opgaven.
3. Hvad er **den bedste** algoritme som løser opgaven?

Udvikling og vurdering af algoritmer

Punkt 1 Find (mindst) én algoritme som løser opgaven: Kræver ideer, tænkearbejde, erfaring, og en værktøjskasse af kendte algoritmer generelle algoritmeudviklingsmetoder. Korrekthed af algoritme: ved analyse eller implementation/afprøvning?

Punkt 2 Sammenlign flere algoritmer som løser opgaven: Kræver definition af hvad er kvalitet. Sammenligning: ved analyse eller implementation/afprøvning?

Analyse: Giver høj sikkerhed for korrekthed af metoden/idéen. Sparer implementationsarbejde. Sammenligning upåvirket af: maskine, sprog, programmør, konkrete input. **Afprøvning:** fanger implementationsfejl, belyser andre/flere ting end analysen, .

Punkt 3 Hvad er den bedste algoritme som løser opgaven? Kan *kun* afklares med analyse.

Udvikling og vurdering af algoritmer

DM507/DS814/SE4-DMAD vil have mest fokus på analyse, lidt mindre på implementation og afprøvning.

I alle byggefag analyserer og planlægger man før man bygger (tænk f.eks. storebæltsbro). Din fremtidige chef vil (bør) forlange det!

Målsætning for kurset

DM507 giver dig en værktøjskasse af algoritmer for fundamentale opgaver, samt metoder til at udvikle og analysere nye algoritmer og varianter af eksisterende.



Målsætning for kurset

Regneøvelser og programmeringsprojekter øger din forståelse for værktøjerne og træner dig i brug af værktøjskassen.



Undervejs begejstres du måske også over smarte og elegante ideer i algoritmer og analyser.



Konkret indhold af kurset

Algoritmer:

- ▶ Analyse af algoritmer: korrekthed og køretid
- ▶ Del og hersk algoritmer
- ▶ Grådige algoritmer
- ▶ Dynamisk programmering
- ▶ Sortering
- ▶ Graf-algoritmer
- ▶ Huffman-kodning

Datastrukturer:

- ▶ Ordbøger (søgetræer og hashing)
- ▶ Prioritetskøer (heaps)
- ▶ Disjunkte mængder

Algorithmanalyse

Algoritmeanalyse

Mindstekrav til algoritmer for at løse et problem:

- ▶ Stopper for alle input (aldrig uendelig løkke).
- ▶ Korrekt output når stopper (giver et svar på vores problem).

Kvalitet af algoritmer som opfylder mindstekrav:

- ▶ Hastighed
- ▶ Pladsforbrug
- ▶ Komplexitet af implementation
- ▶ Ekstra egenskaber (problemspecifikke), f.eks. stabilitet af sortering.

For analyse af dette kræves følgende ingredienser: klar beskrivelse af problem og maskine (modeller), en definition af kvalitet, samt en værktøjskasse af analyseredskaber.

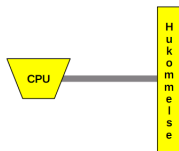
Ingredienser i algoritmeanalyse

I kurset skal vi møde:

- ▶ Model af problem. Individuelt for hvert problem.
- ▶ Model af maskine. Vi bruger RAM-modellen.
- ▶ Mål for ressourceforbrug. Vi fokuserer på tid og plads.
- ▶ Matematiske analyseværktøjer: Løkkeinvarianter, induktion, rekursionsligninger.

Vi uddyber de tre sidste over de næste par sider.

RAM-modellen



- ▶ En CPU
- ▶ En hukommelse (\sim uendeligt array af celler).
- ▶ Et antal basale operationer: *add*, *sub*, *mult*, *shift*, *compare*, *flyt dataelement*, *jump i program* (løkke, forgrening, metodekald). Disse antages alle at tage samme tid.
- ▶ **Tid** for en algoritme: antal basale operationer udført.
- ▶ **Plads** for en algoritme: maks antal optagne hukommelsesceller.

Måle ressourceforbrug

For en givet størrelse n af input er der ofte mange forskellige konkrete input.

Eksempel: for sortering af $n = 8$ tal er følgende nogle af de mulige input:

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

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

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

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

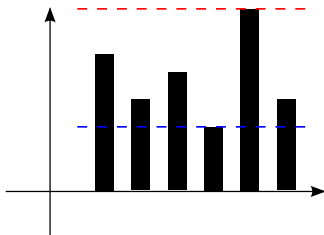
En algoritme har som regel forskelligt ressourceforbrug på hver af disse.

Hvilke input skal vi bruge til at vurdere ressourceforbruget?

Måle ressourceforbrug

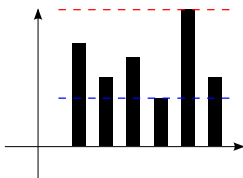
Hvilket input skal vi bruge til at vurdere ressourceforbruget?

- ▶ **Worst case** (max over alle input af størrelse n)
- ▶ Average case (gennemsnit over en fordeling af input af størrelse n)
- ▶ **Best case** (min over alle input af størrelse n)



Køretid for de forskellige input af størrelse n

Worst case ressourceforbrug



Worst case giver **garanti**. Ofte repræsentativ for average case (men nogen gange betydeligt mere pessimistisk).

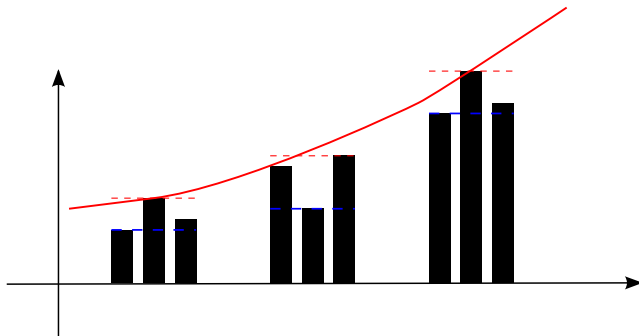
Average case: Hvilken fordeling? Er den realistisk? Ofte svær analyse at gennemføre (matematisk svær).

Best case: Giver ofte ikke så megen relevant information.

Næsten alle analyser i dette kursus er worst case.

Forskellige inputstørrelser

Worstcase køretid er normalt en voksende funktion af inputstørrelsen n :

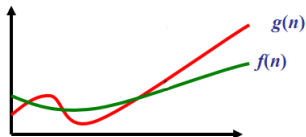


Køretid for de forskellige input af stigende størrelse n

Voksehastighed

Forbruget skal derfor ses som en **funktion** $f(n)$ af inputstørrelsen n .

Vi har derfor brug for at **sammenligne funktioner**. Det relevante mål er **voksehastighed** - en hurtigere voksende funktion vil altid overhale en langsomt voksende funktion når n bliver stor nok. Og for små n er (næsten) alle algoritmer hurtige.



Voksehastighed

Eksempler (stigende voksehastighed):

$$1, \log n, \sqrt{n}, n, n \log n, \\ n\sqrt{n}, n^2, n^3, n^{10}, 2^n$$

Næste gang: mere præcis definition af [asymptotisk voksehastighed](#) og sammenligninger heraf.

Konkret eksempel på algoritmeanalyse

Ombytningsspuslespil . . .