Eva Hadler Vihovde og Harald H. Soleng

Diskret matematikk avslørt

Det er den som styrer datamaskinen

Universitetsforlaget

© Aschehoug AS ved Universitetsforlaget 2026

ISBN 9788215068831 Papirbok
ISBN 9788215068848 PDF-utgave

Materialet er vernet etter åndsverkloven. Kopiering og tilgjengeliggjøring er ikke tillatt uten samtykke fra rettighetshaverne, avtale med Kopinor (www.kopinor.no) eller annen forvaltningsorganisasjon, eller hjemmel i lov. Forbudet gjelder også trening av og annen bruk av materialet i kunstig intelligens, og innebærer et uttrykkelig forbehold mot tekst- og datautvinning etter digitalmarkedsdirektivet artikkel 4.

Henvendelser om denne utgivelsen kan rettes til:

Universitetsforlaget
Postboks 508 Sentrum

0105 Oslo

www.universitetsforlaget.no

Forfatterne har mottatt støtte fra Norsk faglitterær forfatter- og oversetterforening.

Omslag: Ellen Lorenzen
Sats: Harald H. Soleng
Boka er satt med: LuaLaTeX
Trykk og innbinding: Merkur Grafisk AS
Papir: 90g Arctic Matt

INNHOLD

Forord **ix**

1 Introduksjon **1**
1.1 Hva er matematikk? 1
1.2 Hva er diskret matematikk? 3
1.3 Læringsmål 4
1.4 Leserveiledning 5
1.5 Tårnet i Hanoi 6

2 Utsagnslogikk **11**
2.1 Hva er et utsagn? 12
2.2 Symbolbruk og logiske operatorer 13
2.2.1 Ekvivalente utsagn 13
2.2.2 Negasjon 14
2.2.3 Konjunksjon – logisk og 14
2.2.4 Inklusiv disjunksjon – logisk inklusiv eller 15
2.2.5 Unære og binære operatorer 16
2.2.6 Eksklusiv disjunksjon – logisk eksklusiv eller – XOR 16
2.2.7 Logisk implikasjon eller subjunksjon 18
2.2.8 Logisk biimplikasjon eller dobbeltimplikasjon 22
2.2.9 Antall rader i en sannhetsverditabell 23
2.3 Regler om ekvivalente utsagn 23
2.3.1 Lover for negasjon 24
2.3.2 Lover for konjunksjon 24
2.3.3 Lover for inklusiv disjunksjon 25
2.3.4 Kommutative lover 25
2.3.5 Assosiative lover 25

2.3.6 Distributive lover . . . 25
2.3.7 De Morgans lover . . . 26
2.4 Naturlige språk og formelle logiske uttrykk . . . 26
2.5 Boolsk algebra . . . 28
2.5.1 Grunnleggende regneregler . . . 28
2.5.2 Lover for logisk og . . . 28
2.5.3 Lover for logisk eller . . . 28
2.5.4 Kommutative lover . . . 29
2.5.5 Assosiative lover . . . 29
2.5.6 Distributive lover . . . 29
2.5.7 De Morgans lover . . . 29
2.5.8 Absorbsjonslover . . . 29
2.6 Logiske kretser . . . 30
2.7 Logiske operatorer i programmeringsspråk . . . 30
2.8 Øvelsesoppgaver . . . 32

3 Predikatlogikk **37**
3.1 Utsagnsfunksjoner . . . 39
3.2 Kvantorer . . . 40
3.2.1 Kvantorutsagn . . . 41
3.2.2 Negasjon av kvantorutsagn . . . 42
3.2.3 Nøstede kvantorer . . . 44
3.2.4 Negasjon av nøstede kvantorutsagn . . . 46
3.3 Java-eksempel . . . 47
3.4 Øvelsesoppgaver . . . 49

4 Mengdelære **53**
4.1 Mengder . . . 54
4.1.1 Notasjon for mengder . . . 54
4.1.2 Definering av mengder . . . 55
4.1.3 Likhet mellom mengder . . . 57
4.1.4 Venndiagram . . . 58
4.1.5 Delmengder og ekte delmengder . . . 59
4.1.6 Kardinalitet: størrelsen til en mengde . . . 60
4.1.7 Potensmengder . . . 62
4.1.8 Det kartesiske produktet mellom to mengder . . . 63
4.2 Operasjoner på mengder . . . 64
4.2.1 Snitt . . . 64
4.2.2 Union . . . 66
4.2.3 Differansen mellom to mengder . . . 68
4.2.4 Symmetrisk differanse . . . 69
4.2.5 Komplement . . . 71

4.3 Tre mengder . . . 73
4.3.1 Ulike måter å beskrive en mengde på . . . 74
4.3.2 Hvordan avgjøre ekvivalens mellom mengder . . . 75
4.3.3 Antallet elementer i unionen av tre mengder . . . 76
4.4 Mengdeidentiteter . . . 77
4.5 Mange mengder . . . 81
4.6 Venndiagrammer og telleoppgaver . . . 81
4.7 Øvelsesoppgaver . . . 86

5 Følger og rekker **93**
5.1 Følger . . . 95
5.1.1 Aritmetiske følger . . . 96
5.1.2 Geometriske følger . . . 98
5.1.3 Rekurrente følger . . . 100
5.1.4 Noen viktige følger . . . 101
5.2 Rekker . . . 102
5.2.1 Summetegnet . . . 102
5.2.2 Aritmetiske rekker . . . 105
5.2.3 Geometriske rekker . . . 108
5.3 Øvelsesoppgaver . . . 114

6 Differensligninger **117**
6.1 Rekursiv definisjon av en tallfølge . . . 118
6.2 Differensligninger . . . 119
6.2.1 Lineær differensligning av **k**-te orden . . . 120
6.2.2 Startbetingelsene . . . 120
6.2.3 Løsning av differensligninger . . . 121
6.2.4 Løsning av andreordens differensligninger . . . 123
6.2.5 Løse den karakteristiske ligningen . . . 125
6.3 Kodeeksempler Java . . . 132
6.3.1 Fibonacci-tallene . . . 132
6.3.2 Hanois tårn . . . 133
6.4 Øvelsesoppgaver . . . 134

7 Bevisteknikk **139**
7.1 Logisk slutning . . . 140
7.2 Direkte og indirekte bevis . . . 142
7.2.1 Direkte bevis . . . 142
7.2.2 Kontrapositivt bevis . . . 143
7.2.3 Bevis ved selvmotsigelse . . . 145
7.2.4 Praktiske råd . . . 148
7.3 Induksjonsbevis . . . 149

7.3.1 Oppbygningen av et induksjonsbevis 150
7.3.2 Bevis for formelen for summen av en aritmetisk rekke 155
7.4 Øvelsesoppgaver 158

8 Tallteori og kongruensregning **161**
8.1 Heltallsdivisjon og rest 162
8.1.1 Sentrale begreper, symboler og operatorer 162
8.1.2 Divisjonsalgoritmen 166
8.1.3 Operatorene **div** og **mod** 167
8.1.4 Divisjon ved hjelp av gjentatt subtraksjon 167
8.1.5 Heltallsdivisjon der dividenden er negativ 168
8.2 Kongruens og modulo-regning 169
8.2.1 Hva menes med kongruens og modulo? 169
8.2.2 Regneregler for kongruenser 173
8.2.3 Tverrsum og gjentatt tverrsum 174
8.2.4 Kongruensligninger 178
8.3 Kontrollsiffer 179
8.3.1 ISBN 180
8.3.2 GTIN 180
8.4 Primtall 181
8.4.1 Aritmetikkens fundamentalteorem 181
8.4.2 Hvordan avgjøre om et heltall er et primtall? 182
8.4.3 Primtallsfaktorisering 182
8.4.4 Eratosthenes sil 183
8.5 Største felles divisor 185
8.5.1 Finne største felles divisor ved hjelp av primtallsfaktorisering 185
8.5.2 Euklids algoritme 187
8.6 Minste felles multiplum 189
8.6.1 Relativt primiske tall 191
8.7 Øvelsesoppgaver 193

9 Tallsystemer **201**
9.1 Posisjonssystemer 202
9.2 Konvertering mellom tall i ulike posisjonssystemer 204
9.2.1 Fra desimale tall til binære, oktale og heksadesimale tall 205
9.2.2 Fra binære til oktale og heksadesimale tall 208
9.2.3 Fra binære til desimale tall 211
9.3 Binære regneoperasjoner 213
9.3.1 Binær addisjon 214
9.4 Heltallsformat og toerkomplement 214

9.4.1 Fortegnsbit . . . 214
9.4.2 Toerkomplement . . . 215
9.5 Øvelsesoppgaver . . . 218

10 Kombinatorikk **223**
10.1 Sentrale begreper og regneoperasjoner . . . 224
10.1.1 Fakultet . . . 225
10.1.2 Binomialkoeffisienter . . . 226
10.1.3 Pascals trekant . . . 228
10.1.4 Binomialteoremet . . . 230
10.1.5 Oppsummering . . . 232
10.2 Opptelling . . . 232
10.2.1 Produktregelen . . . 232
10.2.2 Sumregelen . . . 234
10.2.3 Subtraksjonsregelen . . . 236
10.2.4 Divisjonsregelen . . . 238
10.2.5 Skuffeprinsippet . . . 240
10.2.6 Oppsummering . . . 241
10.3 Permutasjoner og utvalg . . . 241
10.3.1 Permutasjoner . . . 242
10.3.2 Permutasjoner av ulike objekter . . . 243
10.3.3 Ordnede og uordnede utvalg uten tilbakelegging . . . 246
10.3.4 Ordnede og uordnede utvalg med tilbakelegging . . . 252
10.3.5 Oppsummering . . . 256
10.4 Fischer-sjakk . . . 257
10.5 Øvelsesoppgaver . . . 261

11 Generelle funksjoner **271**
11.1 Hva er en funksjon? . . . 272
11.1.1 Ikke alle relasjoner er funksjoner . . . 273
11.1.2 Verdimengde kontra verdiområde . . . 274
11.1.3 Visualisering av funksjoner . . . 275
11.2 Funksjoners egenskaper . . . 276
11.2.1 Injektiv . . . 277
11.2.2 Surjektiv . . . 278
11.2.3 Bijektiv . . . 279
11.3 Sammensetning av funksjoner . . . 281
11.4 Inverse funksjoner . . . 282
11.5 Tallfunksjoner . . . 284
11.5.1 Funksjonen tak, ceil . . . 285
11.5.2 Funksjonen gulv, floor . . . 286
11.5.3 Funksjonen avrund til heltall, round . . . 287

11.5.4 Funksjonsgrafer 288
11.6 Øvelsesoppgaver 290

12 Tallmatriser og logiske matriser **297**
12.1 Matriser 298
12.1.1 Dimensjon og form 299
12.1.2 Generell matrise 299
12.1.3 Kvadratiske matriser 300
12.1.4 Diagonaler 300
12.2 Matrisearitmetikk 301
12.2.1 Addisjon og subtraksjon av matriser 301
12.2.2 Tall multiplisert med en matrise 301
12.2.3 Matrisemultiplikasjon 302
12.2.4 Lover for matrisearitmetikk 304
12.2.5 Enhetsmatrisen og nullmatrisen 304
12.2.6 Den transponerte matrisen 305
12.2.7 Symmetri 307
12.2.8 Determinant 307
12.2.9 Invers matrise 310
12.2.10 Matrisefunksjoner 314
12.3 Logisk matrisearitmetikk 314
12.3.1 Logisk *eller* mellom to logiske matriser 315
12.3.2 Logisk *og* mellom to logiske matriser 315
12.3.3 Logisk *eksklusiv eller* mellom matriser 316
12.3.4 Logisk matrisemultiplikasjon 316
12.3.5 Potenser av logiske matriser 317
12.4 Øvelsesoppgaver 319

13 Grafteori **327**
13.1 Hva er en graf? 329
13.1.1 Enkel graf 330
13.1.2 Multigraf 330
13.1.3 Pseudograf 331
13.1.4 Egenskaper ved grafer og noder 332
13.1.5 Veier og stier 334
13.1.6 Eulerveier og eulerkretser 335
13.1.7 Rettede grafer 340
13.2 Endelig tilstandsmaskin 341
13.3 Øvelsesoppgaver 345

14 Relasjoner **353**
14.1 Hvordan beskrive en relasjon? 354
14.2 Kombinasjoner og sammensetninger av relasjoner 359
14.2.1 Kombinasjoner 359
14.2.2 Sammensetningen av to relasjoner 360
14.2.3 En vei gjennom en relasjonsgraf 362
14.3 Egenskaper til relasjoner 365
14.3.1 Refleksivitet 366
14.3.2 Irrefleksivitet 367
14.3.3 Symmetri 368
14.3.4 Antisymmetri 370
14.3.5 Transitivitet 372
14.3.6 Kategorisering av relasjoner ut fra egenskaper 375
14.4 Tillukning av relasjoner 376
14.5 Oppdeling av mengder i partisjoner 379
14.5.1 Ekvivalensrelasjoner 380
14.5.2 Ekvivalensklasser 382
14.5.3 Partisjoner 384
14.5.4 Partielle ordninger 387
14.6 Hassediagram 390
14.7 Leksikografisk og topologisk sortering 396
14.7.1 Leksikografisk sortering 396
14.7.2 Topologisk sortering 397
14.8 Øvelsesoppgaver 400

15 PageRank-algoritmen **409**
15.1 Egenvektorsentralitet 410
15.2 PageRank-sentralitet 414
15.3 Oppsummering 417

16 Løsningsforslag **419**

Bibliografi **507**

Stikkordregister **509**

FORORD

Vi lever i en tid der kunstig intelligens og store språkmodeller for alvor har gjort sitt inntog i våre liv. Skal vi forstå grunnlaget for datamaskinens oppbygning og virkemåte, så vel som avanserte søkealgoritmer og store språkmodeller, krever det kunnskap og ferdigheter i diskret matematikk.

Diskret matematikk blir ofte kalt datamatematikk, fordi det er denne formen for matematikk som ligger til grunn for all datateknologi, maskinvare så vel som programvare. I tillegg til å danne det matematiske fundamentet for andre datafag, er faget i seg selv et kraftfullt verktøy i løsning av konkrete, matematiske problemer og datarelaterte oppgaver.

Det er skrevet mange lærebøker i diskret matematikk, men de færreste er skrevet på norsk. Målet med boka er å tilby en norsk lærebok som knytter matematikken til de andre fagene i datastudiene, samtidig som den tilgjengeliggjør et vanskelig fagfelt til studenter med ulik realfagsbakgrunn.

Boka tar for seg mange temaer som alle har det til felles at de er relatert til datateknologi. Den er primært skrevet for førsteårs studenter i informasjonsteknologi der diskret matematikk gjerne inngår som obligatorisk fag på 10 studiepoeng. Boka krever imidlertid ikke forkunnskaper utover matematikk på grunnskolenivå, og stoffet vil derfor være tilgjengelig for alle som er interessert i datamaskinens matematiske grunnlag, inkludert elever i videregående skole.

Det er mange relevante temaer som det kunne vært ønskelig å ha med i boka, og andre temaer vi gjerne skulle gått mer i dybden på. Vi har imidlertid prøvd å holde oss innenfor rammen på 10 studiepoeng, selv om vi nok i enkelte kapitler har tøyd grensene noe. Vi håper du finner boka interessant og nyttig. Skulle du komme over feil eller mangler, er vi takknemlige for kommentarer.

Vi vil gjerne takke Universitetsforlaget som har gitt oss muligheten til å skrive denne boka, samt Norsk faglitterær forfatter- og oversetterforening som har gitt oss økonomisk støtte til prosjektet. Videre vil vi takke Hårek Haugerud, Ulf Uttersrud, Kjetil Grønning og Storm Jaran Bruvoll Westlie

for faglige bidrag i arbeidet med boka. Til slutt vil vi rette en spesiell takk til fagkonsulent Eiolf Kaspersen for en meget grundig gjennomgang av boka med tilhørende verdifulle tilbakemeldinger.

Oslo, mai 2026

Eva Hadler Vihovde og *Harald Heimtun Soleng*

KAPITTEL 1

INTRODUKSJON

Denne boka er skrevet for datastudenter og andre som ønsker å lære mer om datamaskinen og programvarens matematiske grunnlag. Boka tar for seg de delene av matematikken som er spesielt relevante for den digitale verden, herunder Googles søkealgoritmer og store språkmodeller som ChatGPT. Innholdet vil gi leseren et solid grunnlag for å studere ulike datafag, som programmering, algoritmer og datastrukturer, operativsystemer og ikke minst kunstig intelligens og maskinlæring. I tillegg får leseren et kraftfullt, matematisk verktøy til løsning av ulike oppgaver og datatekniske problemer. Stoffet vi gjennomgår krever ingen forkunnskaper utover matematikk på grunnskolenivå, men realfagsbakgrunn fra videregående skole eller universitet er selvfølgelig ingen ulempe. En god porsjon nysgjerrighet er heller ikke til hinder.

I dette introduksjonskapittelet vil vi gi en kort presentasjon av fagområdet diskret matematikk, redegjøre for bokas læringsmål og oppbygning, for til slutt å gi en liten smakebit på hva teorien kan brukes til.

1.1 Hva er matematikk?

Matematikk er en viktig del av verden vi lever i og inngår i nesten alt vi omgis av. Huset vi bor i, bilen vi kjører, alt av datautstyr og andre elektroniske hjelpemidler, betalingsløsningen på butikken, all infrastruktur, ja, mer eller mindre alt rundt oss er i ulik grad basert på matematiske beregninger.

Men hva er matematikk? Kort fortalt er matematikk et språk som i likhet med andre språk har ord og symboler som kan settes sammen til setninger, eller uttrykk som vi kaller dem i matematikken. Foruten tallene

har vi symboler som gjør det mulig å utføre mer kompliserte beregninger, som aritmetiske operatorer for addisjon, subtraksjon, multiplikasjon og divisjon. I denne boka skal vi stifte bekjentskap med mange flere symboler som gir oss mulighet til å anvende matematikken på nye områder. Mens betydningsinnholdet i vanlige språk, som norsk og engelsk, kan tolkes på forskjellige måter, er det ikke rom for subjektive tolkninger i matematikken. Der er ord og symboler presist definert og utrykkene må følge strenge regler.

Matematikk er ikke bare et språk, men også et verktøy. På samme måte som vi bruker hammer til å slå inn spiker og motorsag til å felle trær, kan vi bruke matematikk til løse problemer, avdekke sammenhenger og konstruere nye ting. *Oppsummert kan vi si at matematikk er et formelt språk, egnet til å beskrive logiske sammenhenger og sannheter, og et verktøy til å beskrive virkeligheten.*

Selve ordet *matematikk* kommer fra det greske ordet μάθημα (máthēma) som betyr kunnskap. I dag brukes ordet om studier av tall, objekter som kan karakteriseres med størrelse, orientering og form og relasjoner mellom slike objekter. Faget er en forutsetning for moderne vitenskap og teknologi, men er likevel ikke noen nymotens oppfinnelse. Som vi skal se mange eksempler på, har matematikken røtter som går tusenvis av år tilbake i tid.



Figur 1.1 Leirtavle med en tilnærming til kvadratroten av 2.
BC.021354 [YBC 7289], Courtesy of the Yale Peabody Museum, Babylonian Collection.

Sumer var et oldtidsrike som lå mellom elvene Tigris og Eufrat. For mer enn 5000 år siden innførte de et tallsystem med 60 som grunntall (seksagesimalt). Rester av dette finner vi den dag i dag i måten vi regner med tid, der det er 60 sekunder i et minutt og 60 minutter i en time. Babylonerne overtok tallsystemet, og på leirtavlen i figur 1.1 fra mellom år

1800 og 1600 f.Kr. finner vi et enhetskvadrat med diagonalen tegnet inn. Langs den horisontale diagonalen kan vi se noen kileskriftsymboler som betyr 1; 24, 51, 10 i 60-tallssystemet. Dette betyr at de hadde regnet ut kvadratroten av 2

$$\sqrt{2} \approx 1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 1,41421296\ldots \tag{1.1}$$

Dette er riktig med 4 seksagesimale siffer eller omtrent 6 desimale siffer.

1.2 Hva er diskret matematikk?

Matematikk er et stort fagområde med mange ulike disipliner. Vi har algebra, kalkulus og matematisk analyse, anvendt matematikk, statistikk, beslutningsteori, beregningsmatematikk, og ikke minst diskret matematikk. Men hva er diskret matematikk og hva er det som gjøre den diskret? Ordet «diskret» gir oss dessverre ingen intuitiv forståelse. Det er faktisk lettere å forklare hva diskret matematikk *ikke* er. Diskret matematikk er ikke kontinuerlig, men forholder seg til distinkte elementer slik som heltall, sannhetsverdier og tellbare mengder. Mens det på andre anvendelsesområder innen realfag trengs matematikk for kontinuerlige rom, slik det brukes i fysikken til Newton og Einstein, er det først og fremst forståelse for diskret matematikk som kreves innen datateknologi. Følgelig kalles denne typen matematikk ofte datamatematikk. Fra den fysiske verden er vi vant til at rommet kan deles inn i et uendelig antall små biter. Mellom tallet 0 og 1 er det for eksempel uendelig mange reelle tall, og disse er ikke engang tellbare. I datamaskiner og andre digitale systemer finnes det imidlertid kun et endelig antall tilstander, og følgelig kan vi kun representere kontinuerlige mengder som diskretiserte tilnærminger. For eksempel består et TV-bilde av et endelig antall piksler, mens virkeligheten synes å være kontinuerlig.

Diskret matematikk er et paraply-begrep som favner ulike matematiske disipliner som alle er relatert til datateknologi på en eller annen måte. Blant de mange temaene vi gjennomgår i denne boka finner vi

- utsagnslogikk og predikatlogikk
- mengdelære
- generelle funksjoner
- følger og rekker
- tallmatriser og logiske matriser
- algoritmer
- tallteori
- heltallsregning
- tallsystemer

- kongruensregning
- bevisteknikk
- induksjon
- kombinatorikk
- binomialkoeffisienter
- differensligninger
- relasjoner og ordninger
- grafer og trær
- endelig tilstandsmaskiner

Ferdigheter og kunnskaper innen disse områdene vil bidra til å gjøre andre datafag lettere tilgjengelig. Vi kan nevne fag som

- programmering
- algoritmer og datastrukturer
- databaser
- operativsystemer
- kompilatorteknikk
- datasikkerhet
- kryptering
- systemadministrasjon
- testing av programvare
- kunstig intelligens
- maskinlæring

Diskret matematikk er imidlertid i seg selv et kraftfullt verktøy som kan brukes til å løse mange ulike problemer. I denne boka skal vi blant annet bruke matematikken til å besvare spørsmål som

- Hvor mange gyldige passord kan velges på et system?
- Hvor stor sjanse er det for å vinne i Lotto eller andre spill?
- Finnes det en link mellom to datamaskiner i et nettverk?
- Hva er den korteste veien mellom to byer hvis jeg bruker offentlig transport?
- Hvordan blir to tall lagt sammen av en datamaskinen?
- Hvordan blir tall representer binært, oktalt og hexadesimalt?

1.3 Læringsmål

Målet med boka er både å gi leseren et matematisk fundament som kan bidra til en dypere forståelse av ulike datafag.

Fagterminologi
Mye av fagterminologien som brukes i ulike datafag er i stor grad funda-

mentert på diskret matematikk. Gjennom å lære seg relevante begreper og terminologi vil det bli lettere for leseren å lære seg andre datafag.

Problemløsningsverktøy
Gjennom oppgaveløsning skal leseren trenes i å bruke teoriene aktivt i løsning av konkrete datatekniske oppgaver og problemer.

Anvendelsesområder
Bokas mål er å gi en forståelse for hvor de ulike temaene brukes innen datateknologi.

Dypere forståelse
I møte med andre datafag skal leseren kunne gjenkjenne de matematiske teoriene, noe som vil bidra til å gi vedkommende en dypere datafaglig forståelse. I boka legger vi spesielt vekt på sammenhengen mellom diskret matematikk og programmering.

Historisk perspektiv
Ved å trekke linjer fra teorienes opphavsmenn og frem til i dag skal leseren kunne sette matematikken og datateknologien i et historisk perspektiv.

1.4 Leserveiledning

Boka omhandler mange ulike matematiske temaer som ved første øyekast kan fremstå som adskilte. Rekkefølgen temaene presenteres i er ikke tilfeldig. Med unntak av kapittel 1 og 2, bygger hvert kapittel på de foregående. Det betyr at det du tilegner deg av kunnskaper og ferdigheter i ett kapittel får du bruk for i de påfølgende kapitlene. Følgelig bør temaene leses i den rekkefølgen de står. Utformingen av de enkelte kapitlene er gjort ut fra følgende pedagogiske motiver og retningslinjer:

Motivasjon er nøkkelen til all læring
For å motivere leseren starter hvert kapittel med å sette temaet inn i en datafaglig kontekst slik av teorien som skal læres oppleves relevant.

Historisk forankring
Da de matematiske teoriene som ligger til grunn for datateknologien gjerne har lange historiske røtter, setter vi også stoffet inn i et historisk perspektiv, der vi trekker linjer fra opphavspersonen(e) og frem til i dag. Disse avsnittene er på grå bakgrunn.

Fra det konkrete til det abstrakte
I gjennomgåelsen av fagstoffet beveger boka seg fra det konkrete og gjenkjennbare til det generelle og abstrakte. Først presenteres et eller flere kon-

krete, gjenkjennelige eksempler, deretter gjennomgås teorien, og til slutt innarbeides den ytterligere gjennom eksempler og oppgaver.

Øvelse gjør mester
Å forstå matematisk teori er en ting, å kunne anvende den i praktisk problemløsning er noe annet. For å kunne bruke matematikken som et verktøy i løsning av konkrete problemer, er mengdetrening i oppgaveløsning en dyd av nødvendighet. Boka presenterer derfor en mengde relevante oppgaver som skal sikre leseren tilstrekkelig læringsutbytte.

Knytte diskret matematikk til programmering
Da mange av leserne er datastudenter som også har programmering i fagkretsen, har vi lagt vekt på å knytte matematikken til programmering. I hvert kapittel viser vi hvordan den aktuelle matematiske teorien kan implementeres i et dataprogram. Som programmeringsspråk har vi valgt Java, da det per dags dato er det mest utbredte programmeringsspråket i norske datautdanninger. Boka avsluttes med et praktisk eksempel som viser hvordan diskret matematikk kan brukes til å lage avanserte algoritmer innenfor datateknologi. Her gjennomgår vi en av Googles mest kjente søkealgoritmer, den såkalte PageRank-algoritmen.

Litteraturhenvisninger i teksten er markert med tall i klammeparenteser.

1.5 Tårnet i Hanoi

Avslutningsvis vil vi, som en liten smakebit på diskret matematikk, presentere et matematisk problem knyttet til en gammel legende, Hanois tårn. Legenden forteller at ved jordas begynnelse plasserte skaperen tre høye stolper på en messingplate ved et tempel i Benares, verdens midtpunkt. På en av stolpene var det plassert 64 gullskiver. Den største lå nederst, og så ble platene mindre og mindre oppover søylen. Munkene ved tempelet arbeider dag og natt, og uten stans, med å flytte gullskiver fra den ene søylen og over på de andre etter Brahmas uforanderlige regler

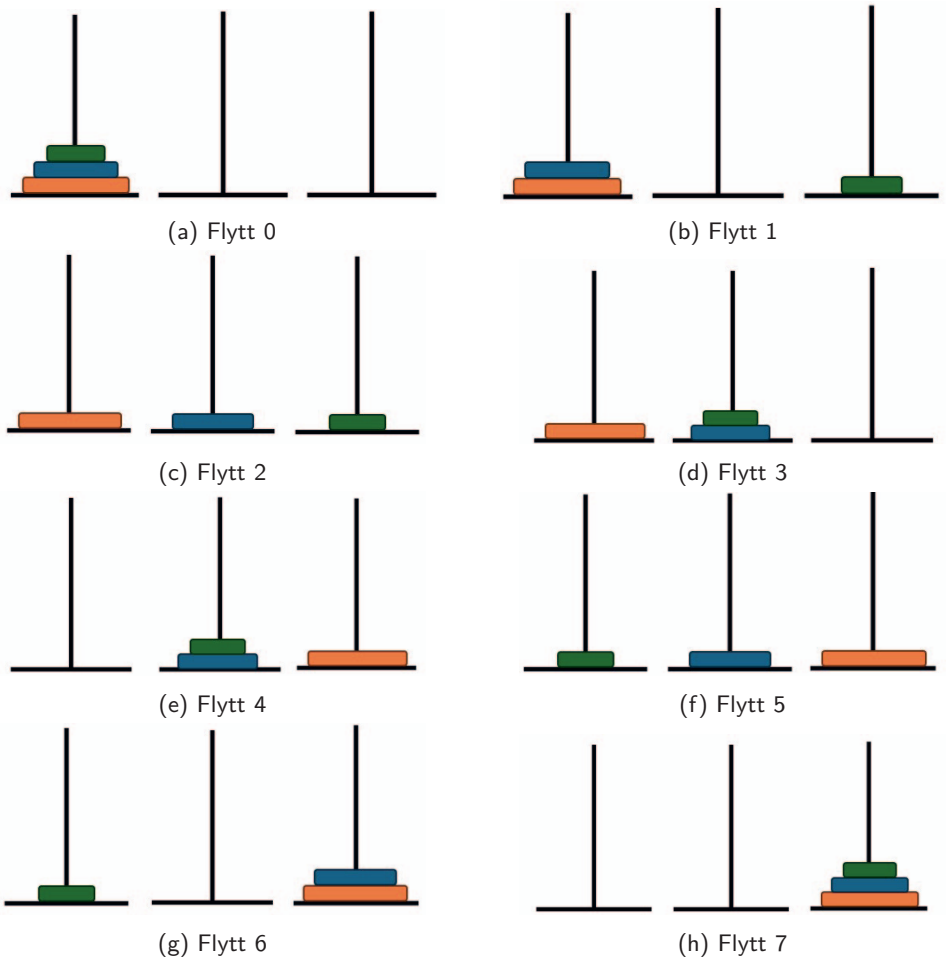
- Kun en plate kan flyttes av gangen
- En plate skal aldri legges oppå en som er mindre

Målet er å få hele tårnet med gullskiver flyttet over til en av de andre søylene. Når dette er gjort, vil verden gå under.

Det var den franske matematikeren Edouard Lucas som i 1883 introduserte legenden i Vesten. Siden den gang er det blitt laget utallige varianter av den. Vi skal nå studere Hanois tårn nærmere og se om vi greier å bestemme hvor mange flytt som kreves for å løse munkenes oppgave. Greier vi det, vil vi kunne estimere hvor lenge det er til verdens undergang.

Når vi skal løse en utfordrende oppgave er det lurt å starte med et enklere, men tilsvarende problem. Vi velger derfor å redusere antall skiver fra 64 til kun 3.

I utgangspunktet ligger alle skivene på søylen til venstre (figur 1.2a). Målet vårt er å flytte dem over til en av de andre søylene, ved å bruke den tredje som hjelpesøyle. For å få flyttet den største skiven til riktig plass nederst på målsøylen, må vi fjerne de to som ligger oppå. Vi kan imidlertid kun flytte én skive av gangen og starter derfor med å flytte den minste til en ledig søyle (figur 1.2b).



Figur 1.2 Miniversjon av Hanois tårn og de syv flyttene som løser problemet.

Skal reglene overholdes, har vi nå kun ett valg for den nest minste skiven; den må flyttes til søylen i midten (figur 1.2c). Den største skiven er nå klar til å flyttes, men før vi kan gjøre det, må vi ha en søyle å flytte

den til. Det er ikke lov å flytte den oppå en som er mindre, så vi må først frigjøre en søyle slik at den største brikken kan plasseres nederst der. Vi flytter derfor den minste skiven oppå den nest minste på midterste søyle (figur 1.2d). Vi kan deretter flytte den største til riktig plass på søylen til høyre (figur 1.2e). Når vi nå er ferdige med den største brikken, er turen kommet til den nest største. Før vi kan flytte denne, må vi fjerne den minste som ligger oppå, som derfor flyttes til den ledige søylen til venstre (figur 1.2f). Dermed kan vi flytte den nest minste skiven til sin endelige plass på søylen til høyre (figur 1.2g). Da gjenstår det kun å flytte den minste skiven til sin endelige plass oppå de to andre på søylen til høyre (figur 1.2h).

Med kun tre skiver er det ikke spesielt vanskelig å løse oppgaven, og vi greide det ved hjelp av syv flytt, som også er det laveste antallet flytt det er mulig å bruke. Hvor mange kreves for å flytte fire skiver? Eller fem? Og ikke minst, hvor mange flytt kreves for å flytte 64 skiver? Det skal vi finne ut ved å bruke det vi har lært av oppgaven over.

La H_n være det minste antallet trekk som trengs for å flytte n skiver. Da vi utførte oppgaven med tre skiver fant vi ut at $H_3 = 7$. La oss spole tilbake og resonnere over hvordan vi løste oppgaven. I utgangspunktet lå alle skivene på søylen til venstre. I vårt tilfelle var $n = 3$.

1. Først fjernet vi de to skivene som lå oppå den nederste og største skiven, dvs. vi flyttet $n - 1 = 2$ skiver til den midterste søyle ved å bruke søylen til høyre som hjelpesøyle. Fordi vi flyttet $n - 1$ skiver blir antall flytt H_{n-1}.

2. Deretter flyttet vi den nederste og største skiven til riktig plass på høyre søyle ved hjelp av ett flytt, dvs. $H_1 = 1$.

3. Så flyttet vi de 2 minste skivene, dvs. vi flyttet $n - 1 = 2$ skiver fra den midterste søylen til riktig plass på høyre søyle ved å bruke venstre søyle som hjelpesøyle. Fordi vi flyttet $n - 1$ brikker blir antall flytt H_{n-1}.

Ved å summere antall flytt i trinnene 1, 2 og 3 får vi en formel for beregning av det minste antall flytt som kreves for å løse oppgaven med n antall skiver.

$$H_n = H_{n-1} + 1 + H_{n-1} = 2H_{n-1} + 1. \tag{1.2}$$

Å flytte null skiver tar selvfølgelig 0 trekk, $H_0 = 0$. Dermed får vi

$$\begin{aligned} H_1 &= 2H_0 + 1 = 2 \cdot 0 + 1 = 1, \\ H_2 &= 2H_1 + 1 = 2 \cdot 1 + 1 = 3, \\ H_3 &= 2H_2 + 1 = 2 \cdot 3 + 1 = 7, \\ H_4 &= 2H_3 + 1 = 2 \cdot 7 + 1 = 15 \end{aligned}$$

og

$$H_5 = 2H_4 + 1 = 2 \cdot 15 + 1 = 31.$$

Legg merke til av hver beregning baserer seg på den foregående. Skal vi beregne H_{64}, dvs. det antall flytt som kreves for å flytte 64 skiver fra en søyle til en annen, må vi utføre 64 utregninger. Det er unektelig litt tungvint. Ser vi derimot nærmere på svarene, ser vi at de følger et mønster

$$\begin{aligned} H_0 &= 0 = 1 - 1 = 2^0 - 1, \\ H_1 &= 1 = 2 - 1 = 2^1 - 1, \\ H_2 &= 3 = 4 - 1 = 2^2 - 1, \\ H_3 &= 7 = 8 - 1 = 2^3 - 1, \\ H_4 &= 15 = 16 - 1 = 2^4 - 1 \end{aligned}$$

og

$$H_5 = 31 = 32 - 1 = 2^5 - 1.$$

Det gir oss formelen

$$H_n = 2^n - 1. \tag{1.3}$$

Vi har nå fått en formel der vi kan regne ut antall trekk direkte. Setter vi $n = 64$ kan vi beregne det minste antall flytt som kreves for løse munkenes oppgave.

$$H_{64} = 2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615. \tag{1.4}$$

Det er derfor ingen grunn til å bekymre seg om verdens undergang riktig ennå. Selv om munkene er i stand til å flytte en skive per sekund og alltid gjør det smarteste trekket, vil hele operasjonen ta $2^{64} - 1$ sekunder eller 585 milliarder år som er omtrent 42 ganger universets nåværende alder!

Som vi har sett er det en overkommelig oppgave å beregne antall flytt som kreves for å løse oppgaven, selv når antall skiver er høyt. Skal vi utføre oppgaven med minimalt antall flytt, må imidlertid skivene flyttes i en bestemt rekkefølge, og når antall skiver er høyt er det nærmest en umulig oppgave for oss mennesker. Ved hjelp av et lite dataprogram kan vi imidlertid få skrevet ut rekkefølgen skivene må flyttes i.

Java-metoden nedenfor er bygget opp på nøyaktig samme måte som vi løste oppgaven over. La oss kalle den venstre søyle for A, den midterste søylen for B, og søylen til høyre for C. Målet blir da å flytte alle skivene fra A til C, med B som hjelpesøyle. Når vi tester programmet med tre skiver, ser vi at rekkefølgen som skrives ut er den samme som den vi brukte da vi løste oppgave. Å teste programmet med $n = 64$ anbefales derimot ikke! Da vil vi oppleve at maskinen «henger seg» fordi det i praksis ikke er mulig

å skrive ut alle de 18 446 744 073 709 551 615 flyttene innenfor den tiden vi har til rådighet.

Kodesnutt 1.1: Tårnet i Hanoi

```
1  public class HanoisTower
2  {
3    public static void main(String[] args)
4    {
5      hanoisTower('A', 'B', 'C', 3);
6    }
7
8    public static void hanoisTower(char A, char B, char C, int n)
9    {
10     if (n == 0) return;
11
12     // Først flytter vi n-1 skiver fra A ttil B,  med  C som
13     // en hjelpesøyle:
14     hanoisTower(A, C, B, n - 1);
15
16     // Så flyyter vi skive n til sin rette plass
17     System.out.println("Disc " + n + " from " + A + " to " + C);
18
19     // Til slutt flytter vi de n-1  skivene fra B til søyle C
20     hanoisTower(B,A,C,n-1);
21   }
22 }
```

Utskrift fra kjøring

```
Disc 1 from A to C
Disc 2 from A to B
Disc 1 from C to B
Disc 3 from A to C
Disc 1 from B to A
Disc 2 from B to C
Disc 1 from A to C
```

Hanois tårn illustrerer hva vi i matematikken kaller *kombinatorisk eksplosjon*, et fenomen som ga startskuddet til kunstig intelligens. Da forskere erfarte at mange programmeringsoppgaver ikke lot seg løse med rå regnekraft, vendte de seg mot den menneskelige hjernen med følgende spørsmål: Kan kunnskap om hjernen hjelpe oss å løse utfordringene kombinatorisk eksplosjon skaper? Svaret på spørsmålet har gitt opphav til fagfelt som kunstig intelligens, nevrale nettverk og maskinlæring.

KAPITTEL 2

UTSAGNSLOGIKK

Et dataprogram er en samling av instruksjoner som maskinen utfører når programmet kjøres. Når vi programmerer må vi bestemme både hvilke instruksjoner som skal utføres og rekkefølgen de skal utførers i. Et alternativ er å utføre dem i den rekkefølgen de står, men vanligvis ønsker vi å bestemme rekkefølge ut fra hva som skjer når programmet kjøres. Dette kan vi få til ved å bruke valg-setninger som, *if-* og *if/else*-setninger eller repetisjonssetninger som *while-*, *do/while-* eller *for*-løkker. Ved å kombinere dem på forskjellige måter kan vi lage ulike algoritmer. Felles for begge typer setninger er at de inneholder logiske betingelser som avgjør om programinstruksjoner skal utføres eller ikke. En logisk betingelse er et utsagn som enten er sant eller usant, og er det grunnleggende elementet i utsagnslogikk. For å skrive dataprogrammer som utfører nøyaktig det vi ønsker at det skal utføre i enhver situasjon, trenger vi å lære utsagnslogikk.

George Boole (1815–1864) var skomakersønn, og gjorde seg tidlig bemerket for sine intellektuelle evner. Allerede som 14-åring ble han en lokal kjendis etter å ha oversatt et gresk dikt som han fikk publisert i lokalavisen. Han hadde ikke råd til å være fulltidsstudent, men lærte seg avansert matematikk ved siden av arbeidet som lærer. Etter å ha publisert flittig i *Cambridge Mathematical Journal* ble han en professor ved Queen's College i Cork [1]. George Boole skapte det vi kaller boolsk algebra eller utsagnslogikk. I over 2000 år hadde logikk og matematikk utviklet seg nærmest adskilt, men med boolsk algebra kan man regne med logiske utsagn på en måte som ligner på normal matematikk.



George Boole
Foto: DEA / BIBLIOTECA AMBROSIANA / Contributor / Getty Images

I 1855 giftet han seg med Mary Everest. Paret fikk fem døtre, hvorav Lucy Everest Boole ble den første kvinnelige kjemiprofessoren i England og Alicia Boole Stott gjorde viktige bidrag i firedimensjonal geometri.

2.1 Hva er et utsagn?

Da denne delen av matematikken heter utsagnslogikk, må vi ha klart for oss hva et utsagn er. Vi har brukt ord som logiske uttrykk og logiske betingelser, men hva er et utsagn? Et utsagn er kort og godt en setning som uttrykker noe vi kan avgjøre om er sant eller ikke. En setning vi kan avgjøre sannhetsverdien av kaller vi gjerne for en *påstand* eller *proposisjon*.

Definisjon 2.1 Utsagn
Et *utsagn* er en erklærende setning som er *sann* eller *usann*.

Kan vi ikke avgjøre sannheten av setningen, er det heller ikke et utsagn. La oss se på noen konkrete eksempler.

Eksempel 2.1 Setninger og utsagn

Avgjør om følgende setninger er utsagn, og bestem eventuelt sannhetsverdien.

1. *Oslo er Norges største by.*
   Setningen er et utsagn dersom begrepene «størst» og «by» er definert. Mener vi størst i innbyggertall, er utsagnet sant. Mener vi størst i bykommunens areal, må Oslo se seg slått av Rana, og utsagnet er usant.
2. *Trikken er aldri i rute.*
   Setningen er et utsagn. Siden det faktisk hender at trikken er i rute, er dette et usant utsagn.
3. $1 + 1 = 2$.
   Setningen er et utsagn som påstår at det som står på venstresiden av likhetstegnet er lik det som står på høyresiden av det. Siden venstresiden er lik høyresiden er dette et sant utsagn.
4. $1 + 1 = 5$.
   Dette er også et utsagn, men denne gangen er venstresiden forskjellig fra høyresiden, slik at dette er et usant utsagn.
5. *Hvor mye er klokka?* Denne setningen er et spørsmål som vi ikke kan avgjøre sannhetsverdien av. Et spørsmål er ikke et utsagn.
6. $x + 1 = 2$. Dette er ikke et utsagn, fordi setningen er en ligning der sannhetsverdien er avhengig av hvilken verdi x har.
7. *I dag er det dårlig vær.* Dette er et utsagn bare hvis begrepet «dårlig vær» er definert.