MAT1030 – Diskret Matematikk

Forelesning 18: Generell rekursjon og induksjon

Dag Normann

Matematisk Institutt, Universitetet i Oslo

17. mars 2010

(Sist oppdatert: 2010-03-17 12:53)

Forelesning 18

Rekurrenslikninger

- Forrige uke så vi på rekurrenslikninger.
- En rekurrenslikning er en funksjonslikning på formen

$$at(n) + bt(n-1) + ct(n-2) = 0$$

hvor en løsning er en funksjon
$F : \mathbb{N} \to \mathbb{R}$ slik at

$$aF(n) + bF(n-1) + cF(n-2) = 0$$

for alle $n \geqslant 3$ (eller for alle n hvor n, $n-1$ og $n-2$ er i definisjonsområdet til F, når vi vil bruke maskineriet vårt i en mer generell situasjon).

- Den karakteristiske likningen er da

$$ax^2 + bx + c = 0$$

Rekurrenslikninger

- Hvis r og s er to forskjellige løsninger av den karakteristiske likningen, er den generelle løsningen av rekurrenslikningen

$$F(n) = Ar^n + Bs^n$$

  hvor A og B er vilkårlige reelle tall.
- Hvis den karakteristiske likningen bare har en løsning r, er den generelle løsningen av rekurrenslikningen

$$(A + Bn)r^n.$$

Rekurrenslikninger

- Hvis vi i tillegg har krav om at $t(1) = a$ og $t(2) = b$, kan vi bestemme A og B i den generelle løsningen ved å sette inn for $n = 1$ og $n = 2$ i den generelle løsningen, og løse likningene mhp A og B.
- Dette vil alltid fungere.
- Boka ser bare på tilfellet med initialbetingelser på $t(1)$ og $t(2)$.
- Hadde vi satt krav til $t(17)$ og $t(256)$, eller til t-verdien for to andre, forskjellige tall, kunne vi fortsatt bestemt A og B fra den informasjonen.
- Initialbetingelser for $n = 0$ og $n = 1$ kan gi den enkleste regningen.
- Dette skal vi se nærmere på om en stund.

Rekurrenslikninger

- Hvis vi har en rekurrenslikning

$$at(n) + bt(n-1) + ct(n-2) = 0$$

  hvor $a = 0$ eller $c = 0$, er likningen strengt tatt ikke av 2. orden, og vi må være litt forsiktige.
- Dette er nøyaktig den situasjonen hvor vi kan ha at 0 er en rot i den karakteristiske likningen.
- Da vil den generelle løsningen være på formen

$$F(n) = Ar^n$$

  hvor $r \neq 0$ er en rot i den karakteristiske likningen.
- Egentlig kan vi her betrakte den karakteristiske likningen som en førstegradslikning med r som den eneste løsningen.

Rekurrenslikninger

Eksempel

- En litt håpløs måte å sende en kryptert binær sekvens på vil være å sende 10 eller 01 valgt vilkårlig der det skulle stått 1 og 0 der det skulle stått 0.
- Det er da opp til mottageren, som er den eneste som kjenner krypteringsmåten, å liste opp alle mulige opprinnelige meldinger og finne den som gir mening.
- Hva er det maksimale antall $F(n)$ opprinnelige meldinger som kan ligge bak en mottatt bitsekvens av lengde n?

Rekurrenslikninger

Eksempel (Fortsatt)

- Hvis $n = 1$ har vi bare en mulighet, den sendte biten er 0
- Hvis $n = 2$ har vi to muligheter, bitsekvensen representerer 1 eller bitsekvensen representerer 00.
- For $n \geqslant 3$ har vi to muligheter:
  - Siste siffer er 0 og representerer en 0. Det totale antall muligheter i den situasjonen er $F(n-1)$ ettersom resten av meldingen også skal representere en bitsekvens.
  - De siste to sifrene representerer 1. Dette svarer egentlig til $F(n-2)$ muligheter totalt.

Rekurrenslikninger

Eksempel (Fortsatt)

- Svaret på problemet får vi ved å løse rekurrenslikningen

$$t(n) = t(n-1) + t(n-2)$$

  med initialbetingelser $t(1) = 1$ og $t(2) = 2$
- Løsningen er da at $F(n)$ er Fibonaccitall nr. $n+1$, noe som viser at metoden er svært upraktisk.

Rekurrenslikninger

- Vi har vært lojale mot læreboka og latt løsninger av rekurrenslikninger være følger, eller funksjoner definert på $\mathbb{N}$.
- Det er imidlertid ikke noe i veien for at vi ser på løsninger definert på hele $\mathbb{J}$, eller fra 0 og oppover.
- Vi kan tolke likningen

$$t(n) - t(n-1) - t(n-2) = 0$$

  for Fibonacci-følgen som en likning der n varierer over hele $\mathbb{J}$.
- Det ville eksempelvis gitt oss at vi kan finne $F(0)$ fra

$$F(2) - F(1) - F(0) = 0,$$

  hvilket gir $F(0) = 0$.

Rekurrenslikninger

- Vi kan godt fortsette nedover med $F(1) - F(0) - F(-1) = 0$ og finne at $F(-1) = 1$.
- Den praktiske nytten vil være at det ofte er enklere å bestemme løsningen til en rekurrenslikning med initialverdier fra $F(0)$ og $F(1)$ fordi de lineære likningene vil bli penere.
- Hvis s og r er løsninger av den karakteristiske likningen, kan vi finne A og B fra
  - $A + B = F(0)$
  - $Ar + Bs = F(1)$
- Har vi bare en løsning r, er forenklingen ved å gå til $F(0)$ enda større.
  - $A = F(0)$
  - $(A + B)r = F(1)$
- Bruker man rekurrenslikningen til å regne ut $F(0)$ er dette en lovlig måte å løse oppgaver på.

Rekurrenslikninger

Eksempel

- Vi har gitt rekurrenslikningen

$$t(n) - t(n-1) - 2t(n-2) = 0$$

  og skal finne løsningen som tilfredstiller initialbetingelse $F(1) = 3$ og $F(2) = 5$.
- Den karakteristiske likningen er

$$x^2 - x - 2 = 0$$

  som har løsninger $r = 2$ og $s = -1$.

Rekurrenslikninger

Eksempel (fortsatt)

- Den generelle løsningen er derfor

$$F(n) = A \cdot 2^n + B \cdot (-1)^n.$$

- Vi ser at $F(0) = 1$ er en alternativ initialbetingelse ved å se på $F(2) - F(1) - 2F(0) = 0$.
- Da løser vi likningene $A + B = 1$ og $2A - B = 3$ og får $A = \frac{4}{3}$ og $B = -\frac{1}{3}$.
- Det kan være en smaksak hva som er den enkleste metoden i hvert enkelt tilfelle.
- Hvis røttene til den karakteristiske likningen er kompliserte uttrykk med kvadratrøtter, er det normalt enklere å ta utgangspunkt i $F(0)$ og $F(1)$ for å bestemme A og B.

Rekursjon og programmering

- Vi startet innføringen av rekursjon med å gi eksempler på hvordan vi kunne finne pseudokoder som svarer til rekursive konstruksjoner.
- Vi kan minne om at hvis
  - $f(1) = a$
  - $f(n+1) = g(f(n), n)$

  er en rekursiv funksjon, og vi har en pseudokode for g, kan vi erstatte denne pseuokoden (som en del av en større kode) med

  $$x \leftarrow g(i, j)$$

  i betydningen at variabelen x får verdien til f når inputvariablene får verdiene til i og j.
- Det er flere måter vi kan lage en pseudokode for g på, vi skal se på to av dem:

Rekursjon og programmering

Eksempel

1 *Input* n $[n \in \mathbb{N}]$

2 $x \leftarrow a$

3 *Output* x

4 **For** $i = 1$ **to** $n - 1$ **do**

  4.1 $x \leftarrow g(x, i)$

  4.2 *Output* x

Merk

Denne pseudokoden vil skrive ut $f(1), f(2), \ldots, f(n)$ i rekkefølge.

Hvis vi bare vil ha ut $f(n)$ blir koden enda enklere:

Rekursjon og programmering

Eksempel

1 *Input* n $[n \in \mathbb{N}]$
2 $x \leftarrow a$
3 **For** $i = 1$ **to** $n - 1$ **do**
  3.1 $x \leftarrow g(x, i)$
4 *Output* x

Rekursjon og programmering

- Læreboka har et lite avsnitt om plassen til rekursjon i enkelte programmeringsspråk.
- Dette er lesestoff, som ikke blir utdypet på forelesningene.
- Enkelte programmeringsspråk tillater til og med en sterkere form for rekursjon, selvkallende prosedyrer.
- Rekursjon er et spesialtilfelle av selvkallende prosedyrer, hvor vi definerer en funksjon $f(x)$ ved å bruke verdier $f(y)$ for enkelte y avhengige av x.
- En slik definisjon kan lett lede til løkkeberegninger eller uendelige beregninger, uten at det er lett å se hvorfor det er tilfelle.
- Vi skal ikke komme nærmere inn på det etter følgende eksempel som vi har sett før i flere varianter.

Rekursjon og programmering

Eksempel

- Da vi ga eksempler på pseudokoder, ga vi et eksempel på en algoritme som ingen ennå vet om vil terminere for alle verdier på input, og vi ga algoritmen i form av en pseudokode.
- Våre pseudokoder fanger ikke opp muligheten for selvkallende prosedyrer, men hadde vi hatt den muligheten, kunne vi betraktet følgende som en meningsfylt algoritme.

Rekursjon og programmering

Eksempel (Fortsatt)

- La $f(1) = 1$.
- La $f(n) = f(\frac{n}{2}) + 1$ hvis n er et partall.
- La $f(n) = f(3n+1) + 1$ hvis $n > 1$ er et oddetall.
- Vi kan da eksempelvis regne ut

$$f(20) = f(10) + 1 = f(5) + 2 = f(16) + 3$$

$$= f(8) + 4 = f(4) + 5 = f(2) + 6 = f(1) + 7 = 8$$

- f er veldefinert som en *partiell* funksjon som vi ikke vet om er *total*.

Rekursjon og programmering

Oppgave

- Det er ikke helt sant at vi ikke kan bruke pseudokoder til å lage algoritmer som svarer til selvkallende prosedyrer.
- Siden dette ikke er en del av MAT1030-pensum skal vi ikke legge stor vekt på det.
- Den som har lyst, kan imidlertid prøve å lage en pseudokode som beregner funksjonen definert ved
  - $f(n) = n^2$ hvis n kan deles på 3.
  - $f(n) = f(5n + 1)$ hvis n ikke kan deles på 3.
- Vil algoritmen gi et svar uansett hva input er?

Skal du løse denne oppgaven må du sannsynligvis bevise noe ved induksjon, og du må selv kunne formulere det du skal bevise.

Generell induksjon og rekursjon

- Vi har argumentert for at konstruksjoner ved rekursjon virker og at induksjon er en gyldig bevisteknikk.
- Vi har begrunnet dette med at vi kan nå alle naturlige tall ved å starte med 1 og så legge til 1 så mange ganger vi trenger.
- En måte å forklare hvorfor induksjonsbevis er matematisk holdbare argumenter på er følgende:
- Vi har at $\mathbb{N}$ er den minste mengden som oppfyller
  - $1 \in \mathbb{N}$
  - Hvis $x \in \mathbb{N}$ vil $x + 1 \in \mathbb{N}$
- Hvis vi da viser en egenskap P ved induksjon, viser vi at
  - $1 \in \{y : P(y)\}$
  - Hvis $x \in \{y : P(y)\}$ vil $x + 1 \in \{y : P(y)\}$.
- Siden $\mathbb{N}$ var den minste mengden med denne egenskapen, må $\mathbb{N} \subseteq \{y : P(y)\}$,

  eller, som vi sier, $P(x)$ holder for alle $x \in \mathbb{N}$.

Generell induksjon og rekursjon

- I logikk og informatikk (og også innen andre deler av matematikken og i andre fag) ser man definisjoner som likner på vår beskrivelse av $\mathbb{N}$; man beskriver en strukturert mengde ved å si hva som kommer inn som start og hvordan man finner mer komplekse elementer av mengden.
- Innen informatikk og logikk beskriver vi gjerne formelle språk på den måten.
- Det er viktig for de som skal studere f.eks. informatikk videre at man har en god forståelse av induksjon og rekursjon og at man har kjennskap til rekursjon over andre strukturer enn $\mathbb{N}$.

Generell induksjon og rekursjon

- I de følgende eksemplene skal vi anta at vi har et alfabet som består av alle de symbolene vi kan finne på tastaturet til en standard datamaskin (norsk standard om presisjonen er nødvendig), hvor tomrom er å betrakte som et eget symbol. Vi bruker bokstaver i *kursiv* som variable over bokstavene på tastaturet, og vi kan bruke andre bokstaver i kursiv som variable for ord, hvor:

Generell induksjon og rekursjon

- Et ord er en ordnet sekvens av bokstaver, hvor bokstavene er skrevet uten ekstra tegn i mellom.
- Det er vanlig å la e betegne det tomme ordet.
- Vi føyer en bokstav til et ord ved ganske enkelt å skrive det inntil ordet på høyre side.
- Dette kan vi bruke til å gi en induktiv beskrivelse av mengden av ord.

Generell induksjon og rekursjon

Definisjon

Mengden av ord er den minste mengden som oppfyller

- Det tomme ordet e er et ord.
- Hvis w er et ord og $\mathtt{b}$ er en bokstav, er $w\mathtt{b}$ et ord.

Generell induksjon og rekursjon

Merk

- Dette eksemplet virker en smule kunstig, fordi vi ikke oppfatter ord slik, selv om de fleste av oss starter med tom linje, og så skriver en og en bokstav fra venstre mot høyre.
- Det spiller imidlertid en rolle for hvordan ord representeres som data om de sees på som ordnede sekvenser med en gitt lengde eller som bygget opp ved at nye bokstaver legges til.
- Ord, slik vi har definert dem, er et spesialtilfelle av lister.
- En liste oppfattes som oftes som at enten er den den tomme listen e, eller så er den et ordnet par av siste element og resten av listen.
- Mange tenker seg at ytterste element (*hodet*) i en liste står til venstre for resten (*halen*) av listen.
- Programmeringspråket LISP er basert på listerekursjon.

Generell induksjon og rekursjon

Eksempel

Som et eksempel på en rekursivt definisjon av en funksjon på mengden av ord kan vi betrakte PL (Push Left) som virker på et ord w og en bokstav a ved:

- $\mathrm{PL}(e, a) = a$
- $\mathrm{PL}(wb, a) = \mathrm{PL}(w, a)b$

Det som her skjer er at PL tar for seg et ord w og en bokstav a, og skriver bokstaven *foran* ordet, slik at vi får aw.

Egentlig burde vi vist denne egenskapen ved PL ved induksjon, men vi avstår i denne omgangen.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

Følgende regne-eksempel viser hvordan PL virker:

$\mathrm{PL}(\mathrm{aba}, \mathrm{c}) =$

$\mathrm{PL}(\mathrm{ab}, \mathrm{c})\mathrm{a} =$

$\mathrm{PL}(\mathrm{a}, \mathrm{c})\mathrm{ba} =$

$\mathrm{PL}(e, \mathrm{c})\mathrm{aba} =$

caba

Generell induksjon og rekursjon

Eksempel

Med utgangspunkt i eksemplet PL fra forrige side skal vi definere en speilingsfunksjon R ved rekursjon. R vil ta et ord som input og output vil være ordet skrevet baklengs:

- Vi definerer R ved:
  - $R(e) = e$
  - $R(wb) = PL(R(w), b)$

Igjen overlater vi bekreftelsen av at funksjonen virker i henhold til spesifikasjonen til den enkelte.

Vi skal se på et eksempel, og i tillegg gi en oppgave, som skal hjelpe til med dette.

Generell induksjon og rekursjon

Eksempel

Vi skal vise ved et regneeksempel i full detalj at $\mathrm{R}(\mathrm{abc}) = \mathrm{cba}$ slik R og PL er definerte.

Vi vil bare bruke symbolet e når vi ellers måtte skrevet *ingenting*, så $e\mathrm{ab}$ er det samme som ab.

$$
\begin{aligned}
&\mathrm{R}(\mathrm{abc}) = \\
&\mathrm{PL}(\mathrm{R}(\mathrm{ab}), \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{PL}(\mathrm{R}(\mathrm{a}), \mathrm{b}), \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{PL}(\mathrm{PL}(\mathrm{R}(e), \mathrm{a}), \mathrm{b}), \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{PL}(\mathrm{PL}(e, \mathrm{a}), \mathrm{b}), \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{PL}(\mathrm{a}, \mathrm{b}), \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{PL}(e, \mathrm{b})\mathrm{a}, \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{ba}, \mathrm{c}) = \\
&\mathrm{PL}(\mathrm{b}, \mathrm{c})\mathrm{a} = \\
&\mathrm{PL}(e, \mathrm{c})\mathrm{ba} = \\
&\mathrm{cba}.
\end{aligned}
$$

Generell induksjon og rekursjon

Oppgave

a) Ved å bruke den rekursive definisjonen av $\mathtt{PL}$, vis hvordan vi skritt for skritt kan finne verdiene av

- $\mathtt{PL}(e, \mathtt{d})$
- $\mathtt{PL}(\mathtt{a}, \mathtt{d})$
- $\mathtt{PL}(\mathtt{ab}, \mathtt{d})$
- $\mathtt{PL}(\mathtt{aba}, \mathtt{d})$

Husk at *variablene* $\mathtt{a}$ og $\mathtt{b}$ kan stå for hvilken som helst av *bokstavene* a, b og d.

b) Vis, ved å bruke definisjonen av $\mathtt{R}$ og egenskapen til $\mathtt{PL}$ at $\mathtt{R}(\mathtt{abac}) = \mathtt{caba}$.

Generell induksjon og rekursjon

Eksempel

Den siste funksjonen vi skal se på i forbindelse med den induktive definisjonen av mengden av ord er sammensetningsfunksjonen $w * v = wv$.

Denne kan også defineres ved rekursjon som følger:

- $w * e = w$
- $w * (vb) = (w * v)b$

Vi kunne gjort det vanskeligere, nemlig brukt rekursjon på første argument:

- $S(e, v) = v$
- $S(wb, v) = S(w, PL(v, b))$

Hvis vi nå vil vise at $S(w, v) = w * v$ for alle ord w og v, kan vi ha god bruk for induksjon.

Generell induksjon og rekursjon

- Automatateori er den matematiske teorien for studiet av formelle regnemaskiner som tar for seg et inputord, og enten prosesserer et outputord eller svarer på et spørsmål om inputordet.
- Flere av disse formelle maskinene leser inputordet fra venstre mot høyre, og utfører en beregning underveis.
- Eksempler på dette er de såkalte endelige tilstandsmaskinene og pushdownautomater eller stakkautomater.
- Hvordan en slik automat virker på et ord defineres ved rekursjon på oppbyggingen av ordet.
- Vi skal ikke innføre automatateori, men vi skal i eksempler og oppgaver se på et par tilfeller hvor rekursjon på ord kan brukes til å erstatte slike formelle regnemaskiner.
- De kraftigste formelle maskinene fanges ikke opp av ord-rekursjon.

Generell induksjon og rekursjon

Eksempel

- La w være et ord hvor vi vet at bare bokstavene a og b forekommer.
- Vi vil finne en algoritme som avgjør om det er like mange bokstaver av hvert slag, eller om det er et overskudd av den ene.
- Som en hjelpefunksjon definerer vi en funksjon $f(w)$ slik at $f(w)$ er differansen mellom antall a'er og antall b'er i w.
  - $f(e) = 0$
  - $f(wa) = f(w) + 1$
  - $f(wb) = f(w) - 1$

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- Vi kan da regne ut

$$f(aababba) = f(aababb) + 1 = f(aabab) - 1 + 1 = f(aabab)$$

$$= f(aaba) - 1 = f(aab) = f(aa) - 1 = f(a) = f(e) + 1 = 1.$$

Generell induksjon og rekursjon

Eksempel (Fortsatt)

- Siden ord er ordnede sekvenser av symboler, kan vi finne pseudokoder som erstatter ord-rekursjon.
- Vi skal gi en pseudokode for beregning av f fra dette eksemplet.
- Her er det viktig at *input* er et ord hvor bokstavene a og b forekommer, og at *output* er et helt tall.
- Vanligvis må man deklarere typene til variablene som en del av programmet, men det formaliserer vi ikke her.

Generell induksjon og rekursjon

Eksempel (Fortsatt)

1 *Input* n $[n \in \mathbb{N}_0]$

2 *Input* w $[w = v_1 \cdots v_n$ er et ord av lengde $n]$

3 $x \leftarrow 0$

4 **For** $i = 1$ **to** n **do**

  4.1 **If** $v_i = a$ **then**

    4.1.1 $x \leftarrow x + 1$

    else

    4.1.2 $x \leftarrow x - 1$

5 *Output* x

Generell induksjon og rekursjon

Merk

- Hvis vi arbeider med algoritmer som skal virke på ord i et alfabet eller lister av data, vil det være aktuelt å innføre egne kontrollstrukturer for listerekursjon.
- Det fins programmeringsspråk av teoretisk interesse, og også av praktisk betydning, hvor det er enkelt å uttrykke listerekursjon og andre former for generell rekursjon.

Påskeferie

Uansett hva klokken er nå, har vi gått igjennom nok stoff, så det gjenstår bare å si

Lykke til med midtermin til de som skal ta en slik en.

GOD PÅSKE