Representasjon av tall på datamaskin
Kort innføring for MAT-IN1105

Knut Mørken

18. oktober 2020

Det er noen få prinsipper fra den første delen av MAT-INF1100 om tall som det er nyttig for studentene i MAT-IN1105 å kjenne til. Det viktigste er kjennskap til hvilke begrensninger som ligger i måten datamaskiner håndterer tall. På datamaskiner representeres all informasjon, inkluder tall, ved hjelp av 0 og 1. Det betyr at tall må representeres i det binære siffersystemet som bare bruker sifrene 0 og 1. For eksempel er

$$19 = 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 10011_2,$$
$$5/16 = 0\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} = 0.0101_2.$$

I MAT-INF1100 brukte vi en god del tid å finne fram til algoritmer for slike konverteringer og egenskaper ved representasjonene. Dette oppsummerer vi her sammen må noen praktiske konsekvenser for beregninger.

Alt stoffet på disse sidene er hentet fra Kalkulus og fra kompendiet som brukes i MAT-INF1100 og MAT-IN'1105, se `http://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h20/kompendiet/matinf1100.pdf`.

1 Hele tall på datamaskin

Hele tall representeres på datamaskin som hele tall skrevet i binærsystemet. Siden ressursene er endelige må antall siffer begrenses. Elektronikken i en datamaskin er vanligvis bygget for å representere heltall med 64 binære sifre. Det betyr at tall som er større rett og slett ikke kan representeres i datamaskinen.

Faktum 1. *På en maskin som representerer heltall med 64 binære sifre er det minste negative og det største positive heltallet gitt ved*

$$I_{\min 64} = -2^{63} = -9\,223\,372\,036\,854\,775\,808,$$
$$I_{\max 64} = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807.$$

I Python er denne begrensningen skjult slik at om disse grensene overskrides blir det fanget opp av programvare som tolererer større tall. Begrensningen ligger da i de aktuelle ressursene i maskinen, noe som kan variere.

2 Reelle tall

I matematikk starter vi med de naturlige tallene $\mathbb{N} = \{1, 2, 3, \ldots\}$ som vi utvider til de hele tallene $\mathbb{Z}$ ved å legge til 0 og de negative tallene, Neste steg er å introdusere brøkene som gir oss de rasjonale tallene $\mathbb{Q}$. I dagliglivet kommer vi langt med de rasjonale tallene, men faktum er at de etterlater huller på tallinjen. For eksempel kan vi ikke løse ligningen $x^2 - 2 = 0$ hvis vi bare har rasjonale tall til rådighet. Det viser seg at vi på en pen måte kan gjøre nok en utvidelse til de reelle tallene $\mathbb{R}$.

Den formelle definisjonen skal vi la ligge her, men det er viktig å ha en intuitiv forståelse av de reelle tallene.

Observasjon 1. *De reelle tallene består av alle mulige desimaltall der vi tillater uendelig mange desimaler. De reelle tallene fyller ut tallinjen, uten hull.*

Eksempel 1. Her er noen velkjente reelle tall:

$$\frac{1}{4} = 0.25,$$
$$\frac{1}{7} = 0.14285714285714285714\cdots,$$
$$\sqrt{2} = 1.41421356237309504880168872420969807856967187537694807 3\cdots.$$

Legg merke til $1/4$ kan skrives med nøyaktig 2 desimaler, mens $1/7$ og $\sqrt{2}$ krever et uendelig antall desimaler. Desimalene i $1/7$ består av en gruppe på 6 siffer som gjentar seg, mens desimalene i $\sqrt{2}$ ikke gjentar seg i noe systematisk mønster.

De rasjonale tallene kan karakteriseres presist ved at de er de eneste reelle tallene som har desimaler som gjentar seg på samme måte som for 1/7. Lengden på gruppen av desimaler som gjentar seg varierer selvsagt fra tall til tall, for 1/4 er det '0' som gjentar seg. De tallene som ikke er rasjonale kalles de *irrasjonale* tallene og disse har følgelig ingen slik systematisk gjentagelse av sifrene.

3 Reelle tall på datamaskin

Reelle tall representeres også med 64 binære sifre på datamaskin (noen ganger bare med 32 binære sifre). I presentasjonen her er egenskapene ved denne representasjonen for enkelhets skyld tolket inn i det desimale tallsystemet.

Det er to grunnleggende utfordringer ved representasjon av reelle tall: De kan bli veldig store i tallverdi (eventuelt veldig små uten å bli 0) og de kan ha uendelig mange sifre. På en datamaskin må både størrelsen og antall sifre begrenses. Dette kan gjøres enkelt om vi skriver reelle tall på *normalform*.

Observasjon 2 (Normalform). *La a være et reelt tall ulik 0. Da kan a skrives entydig som*

$$a = b \cdot 10^n \tag{1}$$

der b er begrenset av

$$\frac{1}{10} \leq |b| < 1 \tag{2}$$

og n er et heltall. Dette kalles normalformen til a, tallet b kalles signifikanden mens n kalles eksponenten til a. Normalformen til 0 *er* $0 = 0 \cdot 10^0$.

Når et reelt tall a skal representeres i datamaskinen skrives det på normalform og signifikanden b og eksponenten n lagres hver for seg. Elektronikken i de fleste maskiner er bygget slik at b og n kan lagres sammen i en samling på 64 binære sifre.

Faktum 2 (IEEE 64-bits flyttall format). *Reelle tall representeres vanligvis ved at 53 sifre er reservert signifikanden og 11 sifre for eksponenten, begge inkludert fortegn. Dette betyr at signifikanden har omtrent 15–17 signifikante, desimale sifre. Det minste og største negative tallet som kan representeres i maskinen blir da omtrent*

$$F^-_{\min 64} \approx -1.8 \times 10^{308}, \quad F^-_{\max 64} \approx -5 \times 10^{-324}.$$

Det minste og største positive tallet blir omtrent

$$F^+_{\min 64} \approx 5 \times 10^{-324}, \quad F^+_{\max 64} \approx 1.8 \times 10^{308}.$$

Representasjonen som er antydet over er i dag standard på svært mange (nesten alle) datamaskiner og er utarbeidet i regi av Institute of Electrical and Electronics Engineers (IEEE). Den refereres ofte til som IEEE-754.

Legg merke til at et flyttall (reelt tall på datamaskin) representeres ved hjelp av 64 binære sifre som hvert kan være 0 eller 1. Altså er det til sammen $2^{64} \approx 10^{19}$ slike tall. Hva er disse tallene?

Faktum 3. *Ethvert flyttall (reelt tall på datamaskin) a er på formen*

$$a = \frac{m}{2^n} \tag{3}$$

der $m \in \mathbb{Z}$ er et heltall og n er et ikke-negativt heltall. Motsatt, hvis tallet a ikke er på denne formen kan det ikke representeres eksakt i maskinen. Verken m eller n kan bli vilkårlig store så det det totale antallet flyttall er begrenset, totalt omtrent 10^{19} ulike tall.

Faktum 3 viser at det er svært store begrensninger i hvilke tall vi kan representere på datamaskinen, noe som betyr at de fleste reelle tall må tilnærmes med et flyttall. Noen tall som kan representeres helt eksakt er $1/2$, $29/16$, $401/2048$. Noen tall som ikke kan representeres eksakt er $1/10$, $8/5$, $1/3$, $\sqrt{2}$. Legg spesielt merke til tall som 0.1 og 0.28 ikke kan representeres eksakt på datamaskin.

4 Feil ved representasjon av reelle tall

Det ovenstående betyr at vi ofte får en liten feil når et reelt tall skal representeres på datamaskin. For eksempel viser det seg at 0.1 skrevet med 21 desimaler (i Python) er 0.10000000000000005551, det vil si at feilen er omtrent $5 \cdot 10^{-18}$.

Generelt gjelder følgende.

Lemma 1. *La a være et tall forskjellig fra 0 som ligger innenfor verdiområdet til 64-bits flyttall. Hvis a tilnærmes med det nærmeste flyttallet $\tilde{a}$ er den relative*

feilen begrenset av

$$\frac{|a-\tilde{a}|}{|a|} \leq 2^{-53} \approx 1.1 \cdot 10^{-16}. \tag{4}$$

Feilen som er angitt i (4) angir det verst tenkelige tilfellet, så som regel vil feilen være mindre. En annen måte å uttrykke dette er ved å si at

$$\frac{a-\tilde{a}}{a} = \epsilon$$

der $|\epsilon| \leq 2^{-53}$. Hvis for eksempel a er 0.5 er $\epsilon = 0$ mens hvis $a = 0.1$ vet vi fra eksempelet over at $\epsilon \approx 5 \cdot 10^{-18}$.

Det mest kritiske med representasjon av reelle tall på datamaskin er ikke begrensningene i størrelse, det er svært sjelden det skaper problemer