Fellesnevner («minste felles multiplum», «least common denominator»)

Når vi har bøker som skal legges sammen og de har forskjellige nevnere, så vet dere at vi må finnes fellesnevneren, FN. Vi faktoriserer[1] alle nevnerne og finner hva de har felles, f.eks.

$$\frac{1}{4}+\frac{5}{6}+\frac{1}{12}$$

Vi faktoriserer:

1) Kopierer et totall ned til FN og stryker et totall i de andre: 4=~~2~~·2 6=~~2~~·3 12=~~2~~·2·3 FN: 2	2) Kopierer neste totall ned til FN og stryker et totall i de andre: 4=~~2·2~~ 6=~~2~~·3 12=~~2·2~~·3 FN: 2·2	3) Kopierer tretallet ned til FN og stryker et tretall i de andre: 4=~~2·2~~ 6=~~2·3~~ 12=~~2·2·3~~ FN: 2·2·3	4) Ingen flere faktorer å kopiere, altså ferdig: 4=~~2·2~~ 6=~~2·3~~ 12=~~2·2·3~~ FN: 2·2·3=12

Det er nå lett å se at nevner 4 mangler faktor 3 og brøken må multipliseres med $\frac{3}{3}$. Brøken endrer jo ikke verdi ved å multipliseres med $\frac{3}{3}=1$. Tilsvarende mangler nevner 6 faktor 2 og må multipliseres med $\frac{2}{2}$. Nevner 12 mangler ingenting.

Da kan vi sette alle brøken på felles nevner 12 ved å multiplisere med henholdsvis $\frac{3}{3}$ og $\frac{2}{2}$. Vi får

$$\frac{1}{4}+\frac{5}{6}+\frac{1}{12}=\frac{1\cdot3}{4\cdot3}+\frac{5\cdot2}{6\cdot2}+\frac{1}{12}=\frac{3}{12}+\frac{10}{6}+\frac{1}{12}=\frac{14}{12}=\frac{7}{6}$$

Vi gjør akkurat det samme selv om det finnes bokstaver i brøkene, f.eks.

$$\frac{1}{4x}+\frac{5}{6x}+\frac{x}{12}$$

Vi faktoriserer:

4x=2·2·x (mangler faktor 3)
6x=2·3·x (mangler faktor 2)
12=2·2·3 (mangler faktor x)
FN: 2·2·3·x=12x

Da kan vi sette alle brøken på felles nevner 12x og får

$$\frac{1}{4x}+\frac{5}{6x}+\frac{x}{12}=\frac{1\cdot3}{4x\cdot3}+\frac{5\cdot2}{6x\cdot2}+\frac{x}{12\cdot x}=\frac{3}{12x}+\frac{10}{12x}+\frac{x}{12x}=\frac{13+x}{12x}$$

Vi gjør på akkurat samme måten selv om det er flere ledd i nevnerne ($x\neq0$ og $(x\neq-1)$:

$$\frac{2}{x+1}+\frac{3}{x}=\frac{2\cdot x}{(x+1)\cdot x}+\frac{3\cdot(x+1)}{x\cdot(x+1)}=\frac{2x+3x+3}{x(x+1)}=\frac{5x+3}{x(x+1)}$$

Fun fact: Datamaskiner faktoriserer

Når dataprogrammer skal faktorisere, jobber de slik:

1. Sett *rest* til tallet som skal faktoriseres. Sett *faktor* til 2.
2. Løkke:
    a. Del *rest* med *faktor*. Går det opp?
        i. Hvis ja, noter *faktor*. Sett ny verdi på *rest* til $\frac{rest}{faktor}$.
3. Gjenta med neste verdi (én større) for *faktor* helt til *rest=1*

[1] Å faktorisere et tall innebærer å finne de *primtallene* som ganget med hverandre gir tallet.

Eksempel:

Faktoriser 12.
Rest=12, faktor = 2
Prøver 12:2 – går opp
En faktor er 2. Rest settes til 12/2=6
Prøver 6 : 2 – går opp.
En faktor er 2. Rest settes til 6/2=3
Prøver 3 : 2 – går ikke opp
Prøver 3 : 3 – går opp
En faktor er 3. Rest settes til 3/3=1
Avslutter da resten er 1.
De tallene som var faktorer, var 2, 3 og 3, så 12=2·3·3.

I programmeringsspråket Visual Basic, har jeg skrevet det slik:

```
'Housekeeping
rtbMFN.Clear()
rest = Long.Parse(txtTall.Text) 'hent tallet fra tekstboksen
faktor = 2 'begynn fra 2
'Løkke med output
Do
    If rest Mod faktor = 0 Then 'går opp
        'gi output
        rtbMFN.Text &= faktor & "  "
        rest \= faktor 'gi rest ny verdi, vi har funnet en faktor
        faktor = 1 'begynn igjen (fra 2)
    End If
    faktor += 1 'prøv om neste tall kan være faktor
Loop Until rest = 1
```

Dette kan synes tungvint. Vi «ser» jo ofte bedre enn dette. Bl.a. kan vi hoppe fra primtall til primtall når vi prøver delingen. Men datamaskiner må ha det inn med teskjeer. Det gjør ikke så mye, for de regner så raskt.

Du kan finne faktorer på nettsiden http://acme.com/software/factor/ og http://wims.unice.fr/~wims/wims.cgi (Den siste finner også faktorer for polynomer som f.eks. x^2+6x+9.)

Mer om primtall

Oppgave 2: Hva er et primtall? Definer!

Fundamentalsetningen sier at alle naturlige tall som er større enn 1 og som selv ikke er et primtall, kan faktoriseres slik at alle faktorene er primtall, og denne faktoriseringen er entydig (når vi ser bort fra rekkefølgen av faktorene siden multiplikasjon er kommutativ).

Primtallene er svært nyttige for kryptering. Det viser seg nemlig at når to primtall ganges med hverandre, blir produktet et tall som kun er delelig med de to primtallene. F.eks. er 13·7=91 som bare kan deles med 7 og med 13. Det betyr at hvis man ganger sammen to meget store primtall – gjerne flere hundre siffer – får man et tall som er meget vanskelig å faktorisere. Det utnyttes i kryptering, der et slikt produkt benyttes som «nøkkel».

Fun fact: Kan alle heltall større enn to skrives som summen av tre primtall? Dette er bare en teori fremsatt av Goldbach og kalles derfor *Goldbachs formodning*[2]. Den som beviser dette vil bli berømt!

[2] Senere er formodningen delt i to: Alle partall større enn fire er summen av to primtall, mens alle oddetall større enn fem er summen av tre primtall. Goldbach regnet også tallet 1 som et primtall – det er senere forlatt. Det finnes mange påståtte bevis, men ingen av dem er foreløpig akseptert av matematikerne.

Det er tid- og arbeidskrevende å finne primtall. Man bruker i praksis datamaskiner til dette. Grekeren Eratosthenes fant en enkel metode kalt *Eratosthenes sil* et par hundre år før vår tidsregning. Den er slik:

1. Sett opp alle naturlige tall fra 2 opp til n
2. Begynne med det første tallet som ikke er strøket over. Det er et primtall.
3. Stryk ut alle tall som er multipler av dette tallet (dvs. delelig med tallet)
4. Fortsett fra pkt. 2 inntil neste tall som skal prøves er minst $\sqrt{n}$.
5. De tallene som ikke da er strøket ut, er primtallene opp til n.

Her har jeg satt opp alle tallene opp til 20. Så har jeg strøket ut alle tall som er delelig med 2. Nå er jeg klar til å fortsette med 3 som er det første tallet som ikke er strøket ut. Deretter vil 5 være neste tall osv.

	2	*3*	~~*4*~~	*5*
~~*6*~~	*7*	~~*8*~~	*9*	~~*10*~~
11	~~*12*~~	*13*	~~*14*~~	*15*
~~*16*~~	*17*	~~*18*~~	*19*	~~*20*~~

Siden $\sqrt{20} = 4{,}47$ kan vi stoppe allerede når vi har gjort ferdig multiplene av 3 fordi 5≥4,47. Da ser tabellen slik ut:

	2	*3*	~~*4*~~	*5*
~~*6*~~	*7*	~~*8*~~	~~*9*~~	~~*10*~~
11	~~*12*~~	*13*	~~*14*~~	~~*15*~~
~~*16*~~	*17*	~~*18*~~	*19*	~~*20*~~

Tallene som står igjen er 2, 3, 5, 7, 11, 13 og 19. Det er primtallene opp til 20.

Oppgave 3: Prøv selv med Eratosthenes sil å finne alle primtall opp til 100.

Det er en morsom animasjon av dette på Wikipedia: https://no.wikipedia.org/wiki/Eratosthenes'_sil

Oppgave 4: Du har funnet at en krypteringsnøkkel er 6 887. Du antar at det er laget ved å multiplisere to primtall med hverandre. Hvilke primtall er det?

Oppgaven er relativt enkelt når de to primtallene er små, men hva med 98 505 049? Selv med en liste over primtall som http://www.miniwebtool.com/list-of-prime-numbers/?to=10000 er dette arbeidskrevende[3]. Kryptering dreier seg ofte om å gjøre arbeidet med dekryptering for en som ikke kjenner nøkkelen så tidkrevende at «hemmeligheten» er utdatert før svaret er funnet. Det er også viktig at man kan offentliggjøre en nøkkel for kryptering, men meldingen kan allikevel ikke dekrypteres uten å kjenne (eller finne med mye prøving/feiling) en hemmelig nøkkel. En slik kryptering er en del av det nettlesere benytter når det står «https://» foran nettadressen, f.eks. når man bruker en nettbank.

Det finnes flere alternative metoder for å finne primtall, f.eks. «Sundarams sil» fra 1934 og «Atkins sil» fra 2003. De er raskere, men også mer kompliserte og passer best for dataprogrammer. *Jeg har et fritt program «PrimeSieve» som generer primtall meget raskt. Det bruker en spesiell optimalisering av Eratosthenes sil.*

[3] Det viser seg å være 9901·9949