

Programmering og Problemløsning

Datalogisk Institut, Københavns Universitet

Arbejdsseddel 6 - gruppeopgave

Jon Sparring

7. oktber - 11. oktober.
Afleveringsfrist: fredag d. 11. oktober kl. 17:00.

Emnerne for denne arbejdsseddel er:

- rekursion.

Opgaverne er delt i øve- og afleveringsopgaver. I denne periode skal I arbejde i grupper med jeres afleveringsopgaver. Regler for gruppe- og individuelle afleveringsopgaver er beskrevet i "Noter, links, software m.m." → "Generel information om opgaver".

Øveopgaver

6ø.0 Skriv en funktion, $\text{fac} : \text{n:int} \rightarrow \text{int}$, som udregner fakultetsfunktionen $n! = \prod_{i=1}^n i$ vha. rekursion.

6ø.1 Skriv en funktion, $\text{sum} : \text{n:int} \rightarrow \text{int}$, som udregner summen $\sum_{i=1}^n i$ vha. rekursion. Lav en tabel som i Opgave 3i.0 og sammenlign denne implementation af sum med [while](#)-implementation og `simpleSum`.

6ø.2 Skriv en funktion, $\text{sum} : \text{int list} \rightarrow \text{int}$, som tager en liste af heltal og returnerer summen af alle tallene. Funktionen skal traversere listen vha. rekursion.

6ø.3 Den største fællesnævner mellem 2 heltal, t og n , er det største heltal c , som går op i både t og n med 0 til rest. Euclids algoritme¹ finder den største fællesnævner vha. rekursion:

$$\text{gcd}(t, 0) = t, \quad (1)$$

$$\text{gcd}(t, n) = \text{gcd}(n, t \% n), \quad (2)$$

hvor $\%$ er rest operatoreren (som i F#).

(a) Implementer Euclids algoritme, som den rekursive funktion

¹https://en.wikipedia.org/wiki/Greatest_common_divisor

$\text{gcd} : \text{int} \rightarrow \text{int} \rightarrow \text{int}$

(b) lav en white- og black-box test af den implementerede algoritme,

(c) Lav en håndkøring af algoritmen for $\text{gcd } 8 \ 2$ og $\text{gcd } 2 \ 8$.

6ø.4 Lav dine egen implementering af `List.fold` og `List.foldback` ved brug af rekursion.

6ø.5 Benyt `List.fold` og `List.foldback` og dine egne implementeringer fra Opgave 6ø.4 til at udregne summen af listen $[0 \dots n]$, hvor n er et meget stort tal, og sammenlign tiden, som de fire programmer tager. Diskutér forskellene.

Afleveringsopgaver

I denne opgave skal I regne med kædebrøker (continued fractions)². Kædebrøker er lister af heltal, som repræsenterer reelle tal. Listen er endelig for rationelle og uendelig for irrationelle tal.

En kædebrøk skrives som: $x = [q_0; q_1, q_2, \dots]$, hvilket svarer til tallet,

$$x = q_0 + \frac{1}{q_1 + \frac{1}{q_2 + \dots}} \quad (3)$$

F.eks. listen $[3; 4, 12, 4]$ evaluerer til

$$x = 3 + \frac{1}{4 + \frac{1}{12 + \frac{1}{4}}} \quad (4)$$

$$= 3 + \frac{1}{4 + \frac{1}{12.25}} \quad (5)$$

$$= 3 + \frac{1}{4.081632653} \quad (6)$$

$$= 3.245. \quad (7)$$

Omvendt, for et givet tal x kan kædebrøken $[q_0; q_1, q_2, \dots]$ udregnes ved følgende algoritme: For $x_0 = x$ og $i \geq 0$ udregn $q_i = \lfloor x_i \rfloor$, $r_i = x_i - q_i$ og $x_{i+1} = 1/r_i$ indtil $r_i = 0$. Nedenfor ses en udregning for tallet $x = 3.245$:

i	x_i	q_i	r_i	$1/r_i$
0	3.245	3	0.245	4.081632653...
1	4.081632653...	4	0.081632653	12.25
2	12.25	12	0.25	4
3	4	4	0	-

Resultatet aflæses i anden søjle: $[3; 4, 12, 4]$.

Kædebrøker af heltalsbrøkker t/n er særligt effektive at udregne vha. Euclids algoritme for største fællesnævner. Algoritmen i 6ø.3 regner rekursivt på relationen mellem heltalsdivision og rest: Hvis

²https://en.wikipedia.org/wiki/Continued_fraction

$a = t \text{ div } n$ er heltalsdivision mellem t og n , og $b = t \% n$ er resten efter heltalsdivision, så er $t = an + b$. Man kan nu forestille sig t , n og b som en sekvens af heltal r_i , hvor

$$r_{i-2} = q_i r_{i-1} + r_i, \quad (8)$$

$$r_i = r_{i-2} \% r_{i-1} \quad (\text{rest ved heltalsdivision}), \quad (9)$$

$$q_i = r_{i-2} \text{ div } r_{i-1} \quad (\text{heltalsdivision}), \quad (10)$$

Hvis man starter sekvensen med $r_{-2} = t$ og $r_{-1} = n$ og beregninger resten iterativt indtil $r_{i-1} = 0$, så vil største fællesnævner mellem t og n være lig r_{i-2} , og $[q_0; q_1, \dots, q_j]$ vil være t/n som kædebrøk. Til beregning af største fællesnævner regner algoritmen i 6ø.3 udelukkende på r_i som transformationen $(r_{i-2}, r_{i-1}) \rightarrow (r_{i-1}, r_i) = (r_{i-1}, r_{i-2} \% r_{i-1})$ indtil $(r_{i-2}, r_{i-1}) = (r_{i-2}, 0)$. Tilføjer man beregning af q_i i rekursionen, har man samtidigt beregnet brøkken som kædebrøk. Nedenfor ses en udregning for brøken $649/200$:

i	r_{i-2}	r_{i-1}	$r_i = r_{i-2} \% r_{i-1}$	$q_i = r_{i-2} \text{ div } r_{i-1}$
0	649	200	49	3
1	200	49	4	4
2	49	4	1	12
3	4	1	0	4
4	1	0	-	-

Kædebrøkken aflæses som højre søjle: $[3; 4, 12, 4]$.

Omvendt, approksimationen af en kædebrøk som heltalsbrøken $\frac{t_i}{n_i}, i \geq 0$ fås ved

$$t_i = q_i t_{i-1} + t_{i-2}, \quad (11)$$

$$n_i = q_i n_{i-1} + n_{i-2}, \quad (12)$$

$$t_{-2} = n_{-1} = 0, \quad (13)$$

$$t_{-1} = n_{-2} = 1, \quad (14)$$

Alle approximationerne for $[3; 4, 12, 4]$ er givet ved,

$$\frac{t_0}{n_0} = \frac{q_0 t_{-1} + t_{-2}}{q_0 n_{-1} + n_{-2}} = \frac{3 \cdot 1 + 0}{3 \cdot 0 + 1} = \frac{3}{1} = 3, \quad (15)$$

$$\frac{t_1}{n_1} = \frac{q_1 t_0 + t_{-1}}{q_1 n_0 + n_{-1}} = \frac{4 \cdot 3 + 1}{4 \cdot 1 + 0} = \frac{13}{4} = 3.25, \quad (16)$$

$$\frac{t_2}{n_2} = \frac{q_2 t_1 + t_0}{q_2 n_1 + n_0} = \frac{12 \cdot 13 + 3}{12 \cdot 4 + 1} = \frac{159}{49} = 3.244897959 \dots, \quad (17)$$

$$\frac{t_3}{n_3} = \frac{q_3 t_2 + t_1}{q_3 n_2 + n_1} = \frac{4 \cdot 159 + 13}{4 \cdot 49 + 4} = \frac{649}{200} = 3.245. \quad (18)$$

Bemærk at approksimationen nærmer sig 3.245 når i vokser.

6g.0 Skriv en rekursiv funktion

```
cfrac2float : lst :int list -> float
```

som tager en liste af heltal som kædebrøk og udregner det tilsvarende reelle tal.

6g.1 Skriv en rekursiv funktion

`float2cfrac : x:float -> int list`

som tager et reelt tal og udregner dens repræsentation som kædebrøk.

6g.2 Skriv en rekursiv funktion

`frac2cfrac : t:int -> n:int -> int list`

som tager tæller og nævner i brøken t/n og udregner dens repræsentation som kædebrøk udelukkende ved brug af heltalstyper.

6g.3 Skriv en rekursiv funktion

`cfrac2frac : lst:int list -> i:int -> int * int`

som tager en kædebrøk og et index og returnerer t_i/n_i approximationen som tuplen (t_i, n_i) .

6g.4 Skriv en white- og black-box test af de ovenstående funktioner.

Afleveringen skal bestå af

- en zip-fil

Zip-filen skal indeholde en src mappe og filen README.txt. Mappen skal indeholde fsharp koden, der skal være en fsharp tekstfil per fsharp-opgave, og de skal navngives 6g0.fsx osv. De skal kunne oversættes med fsharpc og den oversatte fil skal kunne køres med mono. Funktioner skal dokumenteres ifølge dokumentationsstandard, og udover selve programteksten skal besvarelserne indtastes som kommentarer i de fsx-filer, de hører til. Filen README.txt skal ganske kort beskrive, hvordan rapporten oversættes til pdf og koden køres.

God fornøjelse.