Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 6 - gruppeopgave

Jon Sporring

9. - 25. oktober.
Afleveringsfrist: onsdag d. 25. oktober kl. 22:00

I denne periode skal I arbejde i grupper. Formålet er at arbejde med:

- Rekursion

Opgaverne er delt i øve- og afleveringsopgaver.

Øveopgaver

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

6ø.1 Skriv en funktion, `sum : n:int -> 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, `sum : int list -> 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 algoritme[1] finder den største fællesnævner vha. rekursion:

$$\gcd(t, 0) = t, \tag{1}$$
$$\gcd(t, n) = \gcd(b, t \;\%\; n), \tag{2}$$

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

(a) Implementer Euclids algoritm, som den rekursive funktion

[1] `https://en.wikipedia.org/wiki/Greatest_common_divisor`

```
gcd : int -> int -> int
```

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

(c) Lav en håndkøring af algoritmen for `gcd 8 2` og `gcd 2 8`.

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

6ø.5 Benyt `List.fold` og `List.foldback` og din egne implementeringer til at udregne summen af listen `[0 .. 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 fortsatte brøker (continued fractions)[2]. Fortsatte brøker er en liste af heltal, som repræsenterer reelle tal. Listen er endelig for rationelle og uendelig for irrationelle tal. Listen skrives ofte som: $x = [q_0; q_1, q_2, \ldots]$, hvilket svarer til tallet,

$$x = q_0 + \cfrac{1}{q_1 + \frac{1}{q_2 + \ldots}}. \tag{3}$$

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

$$\begin{aligned} x &= 3 + \cfrac{1}{4 + \frac{1}{12 + \frac{1}{4}}} && (4)\\ &= 3 + \cfrac{1}{4 + \frac{1}{12.25}} && (5)\\ &= 3 + \frac{1}{4.081632653} && (6)\\ &= 3.245. && (7) \end{aligned}$$

Omvendt, for et givet tal x kan den fortsatte brøk $[q_0; q_1, q_2, \ldots]$ udregnes ved følgende algoritme: Lad $w = \lfloor x \rfloor$ og $r = x - q_0$. Så er den fortsatte brøk for $x = [w; q_1, q_2, \ldots]$, og $1/r = [q_1; q_2, \ldots]$. F.eks. den fortsatte brøk for 3.245 udregnes som:

Trin	Reelle tal	w	r	$1/r$
1	3.245	3	0.245	4.081632653...
2	4.081632653...	4	0.081632653	12.25
3	12.25	12	0.25	4
4	4	4	0	∞

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

Fortsatte brøker af heltalsbrøkker t/n er særligt effektive at udregne vha. Euclids algoritme for største fællesnævner. Algoritmen i 6ø.3 regner rekursivt på:

$$r_{i-2} = q_i r_{i-1} + r_i, \tag{8}$$

[2] `https://en.wikipedia.org/wiki/Continued_fraction`

hvor $r_i = r_{i-2} \% r_{-1}$ og $q_i = \texttt{gcd}\ r_{i-2}\ r_{i-1}$. Algoritmen starter med $r_{-2} = t$ og $r_{-1} = n$. Når $r_i = 0$, så er største fællesnævner r_{i-2}. Relationen mellem algoritmens parametre og rekursionsligningen ses her:

$$
\begin{aligned}
t = r_{-2} &= q_0 r_{-1} + r_0 = q_0 n + t \% n, & \texttt{gcd t n} \quad (9)\\
n = r_{-1} &= q_1 r_0 + r_1 = q_1 (t \% n) + n \% (t \% n), & \texttt{gcd n (t \% n)} \quad (10)\\
&\vdots\\
r_{i-2} &= q_i r_{i-1} + r_i, & \texttt{gcd}\ r_{i-2}\ r_{i-1} \quad (11)\\
&\vdots\\
r_{j-2} &= q_j r_{j-1} + r_j = q_j r_{j-1}, & \texttt{gcd}\ r_{j-2}\ 0 \quad (12)
\end{aligned}
$$

i hvilket tilfælde største fællesnævner er r_{j-2}, og den fortsatte brøk for a/b er $[q_0; q_1, \ldots, q_i, \ldots, q_j]$.

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

$$t_i = q_i t_{i-1} + t_{i-2}, \tag{13}$$
$$n_i = q_i n_{i-1} + n_{i-2}, \tag{14}$$
$$t_{-2} = n_{-1} = 0, \tag{15}$$
$$t_{-1} = n_{-2} = 1, \tag{16}$$

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

$$\frac{t_0}{n_0} = \frac{q_0 t_{-1} + t_{-2}}{q_0 n_{-1} + n_{-2}} = \frac{3 \cdot 1 + 0}{3 \cdot 0 + 1} = \frac{3}{1}, \tag{17}$$
$$\frac{t_1}{n_1} = \frac{q_1 t_0 + t_{-1}}{q_1 n_0 + n_{-1}} = \frac{4 \cdot 3 + 1}{4 \cdot 1 + 0} = \frac{13}{4}, \tag{18}$$
$$\frac{t_2}{n_2} = \frac{q_2 t_1 + t_0}{q_2 n_1 + n_0} = \frac{12 \cdot 13 + 3}{12 \cdot 4 + 1} = \frac{159}{49}, \tag{19}$$
$$\frac{t_3}{n_3} = \frac{q_3 t_2 + t_1}{q_3 n_2 + n_1} = \frac{4 \cdot 159 + 17}{4 \cdot 49 + 4} = \frac{649}{200}. \tag{20}$$

Denne opgave omhandler implementation, dokumentation og afprøvning af de fire ovenstående algoritmer:

6g.0 Skriv en rekursiv funktion

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

som tager en liste af heltal som fortsat brøk 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 fortsat brøk.

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 fortsat brøk udelukkende ved brug af heltalstyper.

6g.3 Skriv en rekursiv funktion

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

som tager en fortsat brøk og et index og returnerer t_i/n_i approximationen som tuplen `(ti, ni)`.

6g.4 Skriv en hvid- og sort-bokstest af de ovenstående funktioner.

Afleveringsopgaven skal afleveres som et antal fsx tekstfiler navngivet efter opgaven, som f.eks. `6g0.fsx`. Tekstfilerne skal kunne oversættes med fsharpc, og resultatet skal kunne køres med mono. Funktioner skal dokumenteres ifølge dokumentationsstandarden, og udover selve programteksten skal besvarelserne indtastes som kommentarer i de fsx-filer, de hører til. Det hele skal samles i en zip fil og uploades på Absalon.

Til øvelserne forventer vi at I arbejder efter følgende skema:

Mandag 9/10: Afslut 5i og start på øvelsesopgaverne fra 6g

Tirsdag 10/10: Arbejd med øvelses- og afleveringsopgaverne

Fredag 13/10: Arbejd med afleveringsopgaverne