Programmering og Modellering (PoM)
Ugeseddel 5 — Uge 40 — Deadline 3/10

Kim Steenstrup Pedersen, Katrine Hommelhoff Jensen, Knud Henriksen,
Mossa Merhi og Hans Jacob T. Stephensen

25. september 2014

1 Plan for ugen

I denne uge præsenteres *funktionsprogrammering*, det vil sige den særlige programmeringsstil, hvor alle beregninger foretages ved sammensætning af funktioner og variable beholder den værdi, de bindes til, og man slipper for at tænke på lagerpladser. Udover dette vil vi fortsætte med emnet talrepræsentationer herunder repræsentation af flydende tal og faldgrupper ved regning med flydende tal.

Forberedelse til forelæsningerne:

Til tirsdag:
Læs Guttag kapitel 5.3 samt noterne om "Funktionsprogrammering".

Til torsdag:
Læs Guttag kapitel 5.3 samt noterne om "Funktionsprogrammering" og om excess N repræsentation at tal.

1.1 Gruppeopgave

Den 3/10 senest klokken 15:00 skal besvarelse af følgende opgave afleveres elektronisk via Absalon. Opgaven skal besvares i grupper bestående af 1 til 3 personer og skal godkendes, for at gruppedeltagerne kan kvalificere sig til den afsluttende tag-hjem eksamen. Opgavebesvarelsen skal oploades via kursushjemmesiden på Absalon (find underpunktet `ugeseddel5` under punktet `Ugesedler og opgaver`). Kildekodefiler ("script"-filer) skal afleveres som "ren tekst", sådan som den dannes af `emacs`, `gedit`, `Notepad`, `TextEdit` eller hvilket redigeringsprogram man nu bruger (*ikke* PDF eller HTML eller RTF eller MS Word-format). Filen skal navngives *efternavn1.efternavn2.efternavn3*.40.py, mens andre filer skal afleveres som en PDF-fil med navnet *efternavn1.efternavn2.efternavn3*.40.pdf.

5g1 I verdensmesterskabsfinalen i speedway (væddeløb for lette motorcykler) er der 16 deltagere, som konkurrerer 4 ad gangen i "heats". Der er i alt 20 heats, og hver kører er med i 5 af dem. Finalen er tilrettelagt således, at hver kører møder enhver anden præcis en gang.

Benævnes kørerne A, B, C, …, O, P, kan finalen repræsenteres af $\mathcal{B}_1$:

```
B1 = [[A,B,C,D],  [A,E,I,M],  [A,F,K,P],  [A,G,L,N],  [A,H,J,O],
      [E,F,G,H],  [B,F,J,N],  [B,E,L,O],  [B,H,K,M],  [B,G,I,P],
      [I,J,K,L],  [C,G,K,O],  [C,H,I,N],  [C,E,J,P],  [C,F,L,M],
      [M,N,O,P],  [D,H,L,P],  [D,G,J,M],  [D,F,I,O],  [D,E,K,N]]
```

En generalisering af denne situation fører til følgende definition: Ved et *(v,k)-bloksystem* vil vi i denne opgave forstå en mængde af delmængder (heatene) udtaget af en mængde V med v elementer (kørerne) på en sådan måde, at

1. Der er k elementer i hver delmængde.

2. For alle valg af to forskellige elementer fra V er de netop sammen i en af delmængderne.

En mængde af delmængder er et *bloksystem*, hvis der findes parametre v og k, for hvilke den er et (v,k)-bloksystem. Navnet kommer af, at delmængderne undertiden også betegnes *blokke*.

$\mathcal{B}_1$ er for eksempel et bloksystem, fordi det er et (16,4)-bloksystem.

Baggrund Hvad vi her har defineret, falder under det, man kalder et *afbalanceret ufuldstændigt bloksystem* (eng.: *balanced incomplete block design* eller *BIBD*):

Et BIBD har tre parametre (v,k,λ), idet man mere almindeligt kræver, at blokkene ikke nødvendigvis skal være forskellige, men at to forskellige elementer fra V altid skal optræde sammen i præcis λ blokke.

Hvad vi ovenfor har kaldt et "bloksystem" fremkommer med andre ord som specialtilfældet $\lambda = 1$.

BIBD'er blev først bragt i anvendelse ved statistisk forsøgsplanlægning, blandt andet ved planteforædling og ved afprøvning af medicinske præparater, men har også anvendelse i forbindelse med blandt andet fejlfindende og fejlrettende koder i datatransmission.

Problem At konstruere et bloksystem med ønskede værdier af v og k er vanskeligt, og vi vil her betragte den simplere opgave at *tjekke*, hvorvidt en forelagt mængde af delmængder nu virkelig også *er* et bloksystem.

$\mathcal{B}_2$, som vises herunder, er for eksempel et bloksystem (med parametre $v = 7$ og $k = 3$), mens hverken $\mathcal{B}_3$ eller $\mathcal{B}_4$ er det: I $\mathcal{B}_3$ er hvert par af de seks elementer ganske vist netop med i en blok — men blokkene er ikke lige lange! I $\mathcal{B}_4$ er hvert par af de seks elementer sammen i en blok — men tre af parrene er med i to blokke!

$\mathcal{B}_2$ = `[[A,B,C], [A,D,G], [A,E,F], [B,D,F], [B,E,G], [C,D,E], [C,F,G]]`

$\mathcal{B}_3$ = `[[a,b], [a,c,f], [a,d,e], [b,c,e], [b,d,f], [c,d], [e,f]]`

$\mathcal{B}_4$ = `[[a,b,c], [a,b,d], [a,e,f], [b,e,f], [c,d,e], [c,d,f]]`

Spørgsmålet om, for hvilke værdier af v og k der overhovedet eksisterer et (v,k)-bloksystem, er vanskeligt og ikke fuldstændig afklaret. (Man kan dog ret let vise, at der ikke eksisterer noget (6,3)-bloksystem.)

$\mathcal{B}_5$ er endnu et eksempel på et bloksystem:

```
B5 = [[0,1,2],  [3,4,5],  [6,7,8],  [0,3,6],  [1,4,7],  [2,5,8],
      [0,4,8],  [1,5,6],  [2,3,7],  [0,5,7],  [1,3,8],  [2,4,6]]
```

Opgave Definer en funktion `is_BS(`*xrr*`)`, som for en liste *xrr* af lister netop returnerer `True`, hvis listen repræsenterer et bloksystem, og ellers `False`.

Opbyg programmet af simple hjælpefunktioner med hver deres veldefinerede overskuelige opgave, og beskriv hovedfunktion og hjælpefunktioner med passende kommentarer og dokumentationsstrenge.

[Vink: Funktionerne fra 5to2 og 5to3 kan måske være nyttige, ligesom skabelonen ved løsning af 2to7 måske kan genbruges.

Python-modulet `set` må *ikke* benyttes.]

1.2 Tirsdagsøvelser

Besvarelser af disse opgaver skal ikke afleveres, men opgaverne forventes løst inden torsdag.

5ti1 Skriv en funktion `remove_duplicates(`xr`)` som tager en liste som argument og returnerer en ny liste der indeholder de samme elementer (eventuelt i en anden rækkefølge) som dem, der ligger i listen xr, men hvor funktionsværdilisten er dubletfri.

5ti2 Definer en funktion `platon(`ntr`)`, der skal kaldes med et argument ntr, som er en liste af tripler af heltal. Funktionens krop skal bruge formatoperatoren (`%`), sådan at der udskrives en linje for hvert tripel i listen. Linjernes indhold fremgår af dette eksempel:

Funktionskaldet

```
platon([(4,4,6), (6,8,12), (8,6,12)])
```

skal bevirke udskriften

```
En regulær  4-side har  4 hjørner og  6 kanter
En regulær  6-side har  8 hjørner og 12 kanter
En regulær  8-side har  6 hjørner og 12 kanter
```

(Bemærk, at formatet bruger *to* skrivepositioner til hvert heltal.)

5ti3 Definer en funktion `union2(`xr`,`yr`)` af to lister, som returnerer en liste bestående af de elementer, som er indeholdt i den ene af de to argumentlister eller i dem begge. Funktionen kan forudsætte, at hver af de to argumentlister består af lutter forskellige elementer, og skal returnere en liste med lutter forskellige elementer. Hvordan funktionen virker, hvis en af (eller begge) argumentlisterne har dubletter, er uden betydning[1].

1.3 Torsdagsøvelser

Besvarelser af disse opgaver skal ikke afleveres, men opgaverne forventes løst inden tirsdag i efterfølgende uge.

5to1 Hvis du ikke allerede har løst opgaverne 3to1, 3to2, og 3to3, så gør dette nu.

5to2 For en liste xrr `= [`xr_0`, ..., `xr_{n-1}`]` af lister skal funktionskaldet `Union(`xrr`)` returnere listen bestående af de elementer, som er indeholdt i en eller flere af listerne xr_i. Det kan forudsættes, at hver af de indgående lister $xr_0, \ldots, xr_{n-1}$ er uden dubletter.

Definer denne funktion `Union()` ved hjælp af `union2()` fra opgave 5ti3 og en af de indbyggede højereordensfunktioner.

5to3 Definer en højereordensfunktion `forAll(`p`,`xr`)` af to formelle parametre til undersøgelse af, hvorvidt alle elementer fra en given liste opfylder et vist prædikat. Et *prædikat* er en funktion, som returnerer en sandhedsværdi.

Funktionskaldet `forAll(`p`,`xr`)` skal returnere `False`, hvis der findes et element x i listen xr, for hvilket $p(x)$ returnerer `False`, og ellers skal `forAll(`p`,`xr`)` returnere `True`.

Hvad skal `forAll(`p`,[])` returnere?

5to4 Definer en højereordensfunktion `transformer(`d`,`g`,`h`)` af tre formelle parametre, sådan at hvis d er en hashtabel og g og h funktioner, vil `transformer(`d`,`g`,`h`)` returnere en hashtabel d', som for hver tilknytning x`:`y i d indeholder tilknytningen $g(x)$`:`$h(y)$.

`transformer({-1: 2, 2: -1}, lambda x: x*x, abs)`
skal for eksempel returnere `{1: 2, 4: 1}`.

[1] Den ønskede funktion kan opfatte sine argumenter som repræsentationer af mængder og skal selv virke som den mængdeteoretiske foreningsoperator. Fra version 2.4 indeholder Python faktisk an klasse `set` til dette formål, og på instanser fra den klasse virker lodret streg (`|`) som foreningsmængdeoperator. Opgaven skal løses uden dette hjælpemiddel, direkte ved arbejde med lister.

Brug `transformer()` med `math.log` som begge de to transformationsfunktioner i forbindelse med `plt.plot()`, `parallelleLister()` og `danHashtabel()`fra ugeseddel 4, og se derved, hvordan de transformerede observationer fra `pattedyr.txt` nogenlunde ligger på linje.

5to5 Indsættes et element x i en liste $[y_0, \ldots, y_{n-1}]$ med n elementer, dannes en ny liste af længde $n+1$, og der er $n+1$ mulige resultater $[y_0, \ldots, y_{i-1}, x, y_i, \ldots, y_{n-1}]$ afhængigt af, hvilken plads x indsættes på ($0 \leq i \leq n$).

Definer en funktion `IndOveralt(`x`,`yr`)` af to formelle parametre, som for et element x og en liste yr af længde n returnerer en liste bestående af de $n+1$ forskellige lister af længde $n+1$, som kan dannes ved indsættelse af x i yr.

Eksempler: `IndOveralt(2,[])` skal returnere `[[2]]`, og `IndOveralt(1,[3,4])` skal returnere
`[[1,3,4],[3,1,4],[3,4,1]]` (eller en af de fem andre lister med elementerne `[1,3,4]`, `[3,1,4]` og `[3,4,1]` i en eller anden rækkefølge).

Prøv dels at løse opgaven ved iteration (med en `for`-løkke og uden rekursive funktionskald), dels i ren funktionsprogrammeringsstil (uden brug af gemmesætninger, men kun med rekursive kald og `return`-sætninger).

[Vink til den funktionelle løsning: Nå yr ikke er tom, dannes `IndOveralt(`x`,`yr`)` ud fra x, yr, yr`[0]` og resultatet af det rekursive kald `IndOveralt(`x`,`yr`[1:])`.]

5to6 Den største fælles divisor (*greatest common divisor*) for to ikke-negative heltal a og b betegnes $\gcd(a,b)$ og har blandt andet disse egenskaber:

$$\begin{array}{rcll} \gcd(2a,2b) & = & 2\gcd(a,b) & \\ \gcd(2a,b) & = & \gcd(a,b) & b \text{ ulige} \\ \gcd(a,2b) & = & \gcd(a,b) & a \text{ ulige} \end{array} \qquad \begin{array}{rcll} \gcd(a,b) & = & \gcd(a-b,b) & \text{for } a \geq b \\ \gcd(a,b) & = & \gcd(a,b-a) & \text{for } a \leq b \\ \gcd(a,0) & = & \gcd(0,a) & = \ a \end{array}$$

Benyt disse egenskaber til i `python` at definere en funktion til effektiv beregning af to ikke-negative heltals største fælles divisor.