DATALOGI 1 (dADS)

OPGAVESAMLING

Opgave 1

a) Skriv den lineæere og den kvadratiske algoritme for maksimal delsum ([Bentley] kap. 7) i TRINE.

b) Find for hver af algoritmerne en formel der i lighed med tabellen side 75 i [Bentley] udtrykker algoritmens udførelsestid i millisekunder (udtrykket *clock* i TRINE kan anvendes til tidsmålingerne).

c) For hvilke værdier af n indhenter TRINE programmerne det langsomme CRAY program?

Opgave 2

Hvor mange rektangulære delmatricer er der i en $n \times n$ matrix?

Opgave 3

Find en invariant for løkken i den lineære algoritme for maksimal delsum ([Bentley] kap. 7) og bevis, at algoritmen er korrekt.

Opgave 4

Betragt det maksimale delsumsproblem i 2 dimensioner (jfr. opgave 7.7.7 i [Bentley]).

a) Skriv en algoritme, der løser dette problem.

b) Hvor mange additioner udfører algoritmen?

c) En moderne datamaskine kan udføre en addition på 10^{-6} sekunder. Hvor lang tid bruger algoritmen på at lægge tal sammen, når $n = 1000$, $n = 10000$, $n = 100000$?

Opgave 5

a) To nyttige prædikater nondec(S) og perm(S, T) defineres som følger (S og T er vektorer)

$\text{nondec}(S) \ : \forall i \in 1..|S| : S.(i-1) \leq S.(i)$

$\text{perm}(S,T) : \forall t \in 0..|T| : |S^{-1}(T.(t))| = |T^{-1}\ (T.(t))| \wedge |T| = |S|$

Forklar i ord, hvad disse to prædikater siger.

b) Skriv følgende prædikater (S, T og U er vektorer)

glat(S)	: "den numeriske forskel på to på hinanden følgende elementer i S er højst 1"
unik(S)	: "elementerne i S er parvis forskellige"
renset(S, T)	: "T indeholder de samme elementer som S, men uden dubletter"
fletning(S, T, U)	: "U er fletningen af S og T"

Opgave 6

a) Skriv metoden til følgende algoritme og bevis, at den er korrekt.

Algoritme: Heltalsdivision
Stimulans: $a, b : a \geq 0, b > 0$
Respons : $q, r : (a = q * b + r) \wedge (0 \leq r < b)$
Metode :

b) I følgende algoritme repræsenterer vektoren $c(0..n)$ koefficienterne i polynomiet

$$p_c(x) = x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0$$

Skriv metoden til følgende algoritme, og bevis at den er korrekt.

Algoritme: Polynomium
Stimulans: $c(0..n)$: Koefficienterne i $p_c(x)$
a: Int
Respons : $r : r = p_c(a)$
Metode :

c) Skriv metoden til følgende algoritme og bevis, at den er korrekt. Metoden må kun udnytte addition, subtraktion, fordobling og halvering (af lige tal).

Algoritme: Multiplikation
Stimulans: $m, n : m, n \geq 0$
Respons : $r : r = m * n$
Metode :

Opgave 7

Betragt en liste X af heltal som i maksimal delsumsproblemet. Beskriv en algoritme der finder den delliste, der har sum tættest på nul.

Hint: Lad H være "hjælpelisten", så $H.(i) = \sum_{j=0}^{i} X.(j)$ og betragt tallene i H sorteret.

Opgave 8

a) Betragt følgende alternative udgave af Binær Søgning.

Algoritme: Alternativ Binær Søgning
Stimulans : $S : \forall i, j \in 0..|S| : i \leq j \Rightarrow S.(i) \leq S.(j)$
m: målelement
Respons : ?
Metode : $\text{lav}, \text{høj} := 0, |S|$
do $\{(S(0..\text{lav}) < m \leq S(\text{høj}..|S|)) \wedge (\text{lav} \leq \text{høj})\}$
$\text{lav} < \text{høj} \rightarrow$
$\text{midt} := (\text{lav} + \text{høj})/2$
if $S.(\text{midt}) < m \rightarrow \text{lav} := \text{midt} + 1$
| $m \leq S.(\text{midt}) \rightarrow \text{høj} := \text{midt}$
fi
od

Hvad kan der siges om variablen høj, hvis

- m er i listen?
- m ikke er i listen?

b) Betragt følgende algoritme

Algoritme: Binær heltalskvadratrod
Stimulans: $n : n \geq 0$
Respons : $r : r^2 \leq n < (r+1)^2$
Metode : $a, b, r := 0, n+1, (a+b)/2$
do $\{(a^2 \leq n < b^2) \wedge (r = (a+b)/2)\}$
$r^2 > n \rightarrow b := r$
$r := (a+b)/2$
| $n \geq (r+1)^2 \rightarrow a := r+1$

$$r := (a + b)/2$$

od

Vis, at algoritmen er korrekt. Hvorfor hedder den *Binær* heltalskvadratrod?

Opgave 9

I denne opgave betegner S en $n \times n$ matrix hvis elementer er heltal, dvs. S er af typen Matrix hvor

$$\textbf{Type } \text{Matrix} = \textbf{List } (\textbf{List}(\text{Int}))$$

Vi kan opfatte S som en funktion

$$S : X \rightarrow \text{ Int}$$

hvor $X = \{(i, j) \mid i, j \in 0..n\}$.

I denne opgave lægges vægt på, at algoritmerne er effektive.

a) Skriv en søgealgoritme, der leder efter et måleelement m i S, og som i givet fald returnerer et indeks for m. Angiv algoritmens udførelsestid.

b) Som a) men nu vides det, at S er *delvist ordnet*, hvilket betyder, at værdien i rækkerne er ikke-aftagende. Sagt mere formelt gælder der

$$\forall i \in 0..n : \forall j \in 1..n : S.(i, j-1) \leq S.(i, j)$$

c) Som a) men nu vides det, at S er *ordnet*, dvs. værdierne i såvel rækker som søjler er ikke-aftagende. Der gælder altså

$$(\forall i \in 0..n : \forall j \in 1..n : S.(i, j-1) \leq S.(i, j)) \wedge$$
$$(\forall j \in 0..n : \forall i \in 1..n : S.(i-1, j) \leq S.(i, j))$$

Hint: start i "et passende" hjørne.

Opgave 10

a) Betragt følgende funktioner:

$$\begin{aligned} f_1(n) &= n^2 \\ f_2(n) &= 1700\, n^2 + 1000\, n \\ f_3(n) &= \begin{cases} n \text{ hvis } n \text{ er ulige} \\ n^3 \text{ hvis } n \text{ er lige} \end{cases} \end{aligned}$$

Angiv for hvert par af funktioner (f_i, f_j), $i, j \in \{1, 2, 3\}$ om $f_i(n) \in O(f_j(n))$ og/eller $f_i(n) \in \Omega(f_j(n))$.

b) Argumenter for at

- $\log n \in O(n^\epsilon)$ for alle $\epsilon > 0$
- $2^n \notin O(n^k)$ for noget $k \geq 0$

c) Orden følgende funktioner efter deres størrelsesorden:

$$n,\ \sqrt{n},\ \log(n),\ \log(\log(n)),\ (\log(n))^2,\ n/\log n,\ (1/3)^n,\ (3/2)^n,\ n\log(n)$$

Opgave 11

Find tidskompleksiteten af følgende tre sætninger, udtrykt ved n:

```
(+ Var i, j, k: Int
   i := 0
   do i<n →
        j := 0
        do j<n →
             C.(i, j) := 0
             k := 0
             do k<n →
                  C.(i, j) := c.(i, j)+A.(i)*B.(k, j)
                  k := k+1
             od
             j := j+1
        od
        i := i+1
   od
+)
```

```
(+ Var i, j, k: Int
   i := 1
   do i<n →
        j := 0
        do j<i →
             k := 2
             do k<n → k := k*k od
             j := j+1
        od
        i := 2*i
   od
+)
```

```
(+ Var i, j: Int
   i := 1
   do i ≤ n →
        if i mod 2 = 1 →
             j := i
             do j ≤ n → j := j+1 od
             j := 1
             do j ≤ i → j := j+1 od
        fi
        i := i+1
   od
+)
```

Opgave 12

Algoritmen Opdel i [P&D] har en tidskompleksitet, der tilhører $\Theta(n)$.

a) Skriv nye metoder til algoritmen, så dens udførelsestid kommer til at tilhøre

- $\Omega(n \log(n))$
- $O(n\sqrt{n})$
- $\Theta(n^2)$

b) Kan du også skrive en metode hvis udførelsestid tilhører $\Theta(n\sqrt{n})$?

Opgave 13

En bunke kan repræsenteres med følgende rekursive type:

Type Bunke=**Prod**(data:Int,left,right:Bunke)

Betragt følgende procedurer, der begge udskriver indholdet af en bunke:

```
Proc Print1(b: Bunke)
    if is(b) →
        write(b.data, eol)
        Print1(b.left)
        Print1(b.right)
    fi
end Print1

Proc Print2[b: Bunke]
    if is(b) →
        write(b.data, eol)
        Print2[b.left]
        Print2[b.right]
    fi
end Print2
```

Hvad er tidskompleksiteten af kaldene Print_1(b) og Print_2[b], hvis b repræsenterer en bunke der indeholder n elementer og $n = 2^h - 1$ for et $h \in \mathbb{N}$? (Dvs. b er et perfekt balanceret træ).

Opgave 14

I denne opgave er f en funktion

$$f : \mathbb{N}_0 \to \mathbb{N}_0$$

for hvilken følgen $x_0, x_1, \ldots$ defineret ved

$$\begin{aligned} x_0 &= 0 \\ x_{i+1} &= f(x_i) \end{aligned}$$

er periodisk, dvs. der findes i og p så

$$x_i = x_{i+p}$$

Det mindste sådanne p kaldes f's periode og betegnes periode(f).

a) Skriv metoden til følgende algoritme.

   Algoritme: Periode
   Stimulans: f: $f : \mathbb{N}_0 \to \mathbb{N}_0$ periodisk funktion
   Respons : p: p = periode(f)
   Metode :

b) Kan du skrive metoden, så dens lagerforbrug er konstant, dvs. uden at gemme de hidtil sete funktionsværdier?

Opgave 15

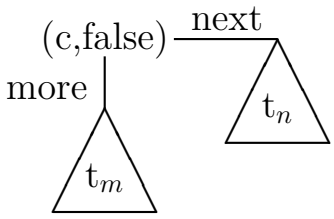
En *trie* er en datastruktur til opbevaring af *ord*, hvor man søger at genbruge fælles præfikser. I TRINE kan vi definere

Type Trie= **Prod**(letter:Char, word:Bool, more,next:Trie)

En knude indeholder et bogstav (letter) samt en angivelse af, hvorvidt det er sidste bogstav i et ord (word). I undertræet more findes fortsættelsen af de ord, der begynder med letter, og i undertræet next er de øvrige ord.

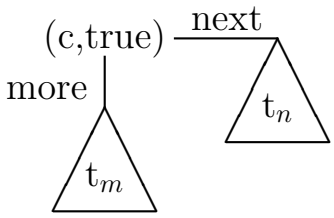
Formelt kan vi definere *words*(t), der er mængden af ord i en trie t:

- hvis t er tom, så er *words*(t)= $\emptyset$
- hvis t er af formen



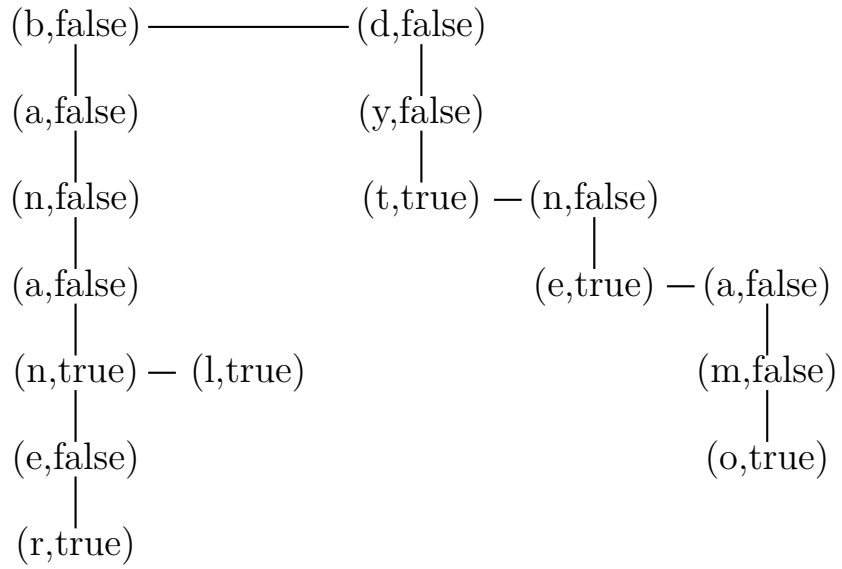
  så er *words*(t) = $\{cw \mid w \in$ *words*(t_m)$\} \cup$ *words*(t_n)

- hvis t er af formen



så er $words(t) = \{c\} \cup \{cw \mid w \in words(t_m)\} \cup words(t_n)$

Hvis t fx. er følgende trie



så er

$words(t) = \{$banan,banal,bananer,dyt,dyne,dynamo$\}$

a) Indsæt ordene *bamse* og *dyd* i ovenstående trie.

b) Skriv en box med procedurer

Proc Init [t: Trie]
Proc Insert [t: Trie] (w: Text)
Proc Member [t: Trie] (w: Text) → (Bool)
Proc Print [t: Trie]

der opfylder specifikationerne

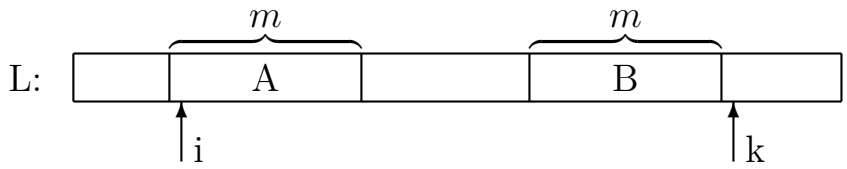
Init[t] **sat** $words(t') = \emptyset$
Insert[t](w) **sat** $words(t') = words(t) \cup \{w\}$
Member[t](w) **sat** $(Member' = (w \in words(t))) \wedge (t' = t)$
Print[t] **sat** $t' = t$.

og hvor Print[t] yderligere udskriver $words(t)$

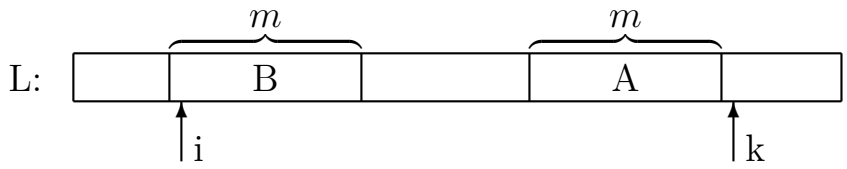
Opgave 16

I denne opgave betragter vi lister, hvis elementer er så store, at vi ikke har råd til at kopiere dem, men kun kan benytte os af *ombytninger*, der jo foregår i konstant tid.

a) Vi er interesserede i at ombytte to sektioner af *samme* længde i en liste således at



laves om til

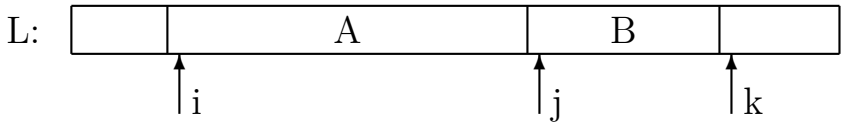


Følgende procedure løser dette problem

```
Proc Flyt[L:Liste](i,k,m: Int)
  (+ Var X: Int
     x:=0
     do x < m →
        L.(i+x) :=: L.(k-m+x)
        x:=x+1
     od
  +)
end Flyt
```

Giv en formel specifikation af proceduren.

b) Vi ønsker nu at ombytte to sammenhængende sektioner af *forskellig* længde i en liste



Beskriv en *rekursiv* procedure Ombyt, der gør dette under anvendelse af proceduren Flyt ovenfor. Den skal tilfredsstille specifikationen

Ombyt[L](i,j,k) **sat** $0 \leq i \leq j \leq k \leq |L| \rightarrow$
$L' = L(0..i) ++ L(j..k) ++ L(i..j) ++ L(k..|L|)$

c) Det påstås, at følgende procedure løser samme opgave som i b)

```
Proc Byt[L:Liste](i,j,k: Int)
  Spejl[L](i,j)
  Spejl[L](i,k)
  Spejl[L](i,i+k-j)
end Byt
```

hvor proceduren Spejl tilfredsstiller specifikationen

Spejl[L](i,j) **sat** $0 \leq i \leq j \leq |L| \rightarrow$
$L' = L(0..i) ++ \Re(L(i..j)) ++ L(j..|L|)$

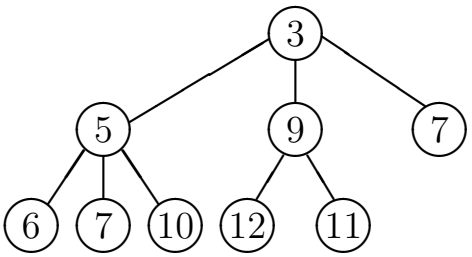
hvor $\Re(X)$ angiver spejlbilledet af X. Er denne påstand korrekt?

Opgave 17

På side 47 i [P&D] påstås det, at prioritetskøer omfatter såvel stakke som køer. Vis dette ved at skitsere nye boxe S og Q, der realiserer hhv. en stak og en kø v.hj.a. prioritetskøen PolyPri side 55.

Opgave 18

En bunke er iflg. noterne et perfekt balanceret *binært* træ, hvor det nederste lag er fyldt så meget som muligt fra venstre mod højre. En *d-bunke* er ligeledes et perfekt balanceret træ (igen eventuelt bortset fra nederste lag), men træets indre knuder har nu d efterfølgere i stedet for blot 2 – en bunke i noternes forstand er således en 2-bunke. Følgende er et eksempel på en 3-bunke.



a) Vis, hvorledes en prioritetskø kan implementeres som en d-bunke (for et vilkårligt $d \geq 2$), og angiv kompleksiteten af prioritetskøens operationer.

b) Antag, at en d-bunke indeholder n elementer. Angiv kompleksiteten af indsættelse og fjernelse når

- $d = 3$
- $d = \log n$
- $d = \sqrt{n}$
- $d = n$

c) Efter hvilke kriterier skal man vælge d i en given anvendelse?

Opgave 19

a) I fremstillingen af hashfunktioner i [P&D] antages det indirekte, at domænet for hashfunktionerne er heltallene. Man har imidlertid også brug for at kunne gemme *ord* i en hashtabel. Diskuter hvordan boxen H skal se ud, hvis Elementtypen er Text i stedet for Int.

b) Betragt håndteringen af *kollisionstlisterne* $D.(i)$ i boxen H. Kunne det gøres mere effektivt? Er det nødvendigt at gøre det bedre?

Opgave 20

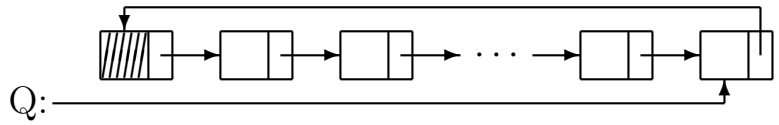
Udvid boxen Search side 68 i [P&D] med to procedurer

Proc Height [T: Tree] → (Int)
Proc Print [T: Tree]

som hhv. returnerer træets højde og udskriver dets elementer i stigende orden.

Opgave 21

a) Skriv en box, der implementerer en kø Q v.hj.a. en liste af følgende form



Listens første element (det skraverede) er et *listehoved*, som ikke indeholder noget køelement. Q peger på listens sidste element.

b) Skriv en box, der implementerer en ordbog v.hj.a. en liste af samme form som i a). Antag, at listens elementer er ordnet i ikke-aftagende orden.

Opgave 22

En *sæk* er en samling elementer, der minder meget om en mængde bortset fra, at i en sæk kan et element forekomme mere end én gang. Vi skelner sække fra mængder ved at bruge symbolerne

$<$	i stedet for	$\{$
$>$	-"-	$\}$
$+$	-"-	$\cup$
$*$	-"-	$\cap$
$-$	-"-	$\setminus$

men vi bruger dog $\in$ og $\emptyset$ med samme betydning som for mængder.

Følgende heltalssæk indeholder således to 3'ere, tre 5'ere og en 1'er

$$S =< 3, 5, 1, 5, 3, 5 >$$

En mængde M (med grundmængde G) er som nævnt side 2 i [P&D] entydigt bestemt af sin karakteristiske funktion.

$$\chi_M : G \to \{\underline{\mathrm{t}}, \underline{\mathrm{f}}\}$$

Det skulle være klart, at en sæk S også er entydigt bestemt ved en karakteristisk funktion

$$\chi_S : G \to \mathbf{N}_0$$

hvor $\chi_S(e)$nu angiver antallet af forekomster af e i S.

a) Giv rimelige definitioner af $*$ og $-$, når $+$ defineres ved

$$S = S_1 + S_2 \Leftrightarrow \forall e \in G : \chi_S(e) = \chi_{S_1}(e) + \chi_{S_2}(e)$$

b) Giv også rimelige definitioner af $\subseteq$ og $|S|$.

c) Hvilke overvejelser ligger der bag din fortolkning af begrebet *rimelighed* i svarene på a) og b)?

Opgave 23

a) En bunke kan bygges ud af n elementer i tid $O(n)$. Hvor hurtigt kan man bygge et søgetræ ud af n elementer?

b) Når en bunke implementeres som en liste, er det let at finde det højreste blad (som er det sidste element i listen).

Hvordan kan man finde det højreste blad i et perfekt balanceret binært træ, der er implementeret som en rekursiv type?

(Vink: Man skal finde den rigtige vej fra roden til det ønskede blad. Denne kan repræsenteres som en følge af 0'er og 1'er, hvor 0 betyder *til venstre* og 1 betyder *til højre*. Overvej hvordan denne følge hænger sammen med antallet af knuder i træet.)

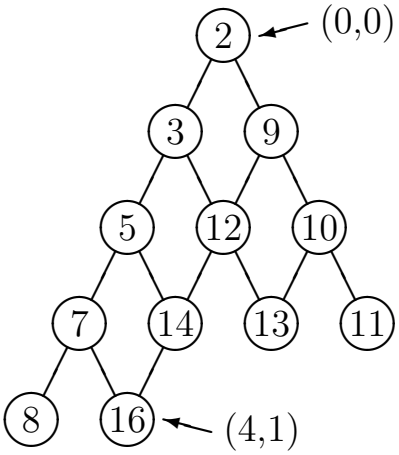
Opgave 24

a) Beskriv en algoritme, der givet n elementer finder det k'te mindste element.

b) Hvordan afhænger algoritmens udførselstid af k?

Opgave 25

Et *tableau* er en perfekt balanceret struktur som den følgende



hvor der i lighed med en bunke gælder, at hvis en knude har sønner, så er disses værdier større end eller lig med knudens værdi. Knuderne kan nummereres, så roden har nummer (0,0) og sønnerne til knuden med nummer (i, j) har numrene $(i + 1, j)$ og $(i + 1, j + 1)$. Et *blad* er en knude uden sønner, og *højden* af et tableau er

$$max\{i \mid (i, j) \text{ er et blad}\}$$

så ovenstående tableau har højde 4.

a) Hvad er højden af et tableau med n knuder?

Betragt følgende datastruktur:

```
Box SP
    Type T = ≪tableau af heltal≫

    Proc Init [t: T]
        ≪Initialiser t≫
    end Init

    Proc Insert [t: T] (e: Int)
        ≪Tilføje til t≫
    end Insert
```

```
    Proc Delete[t: T] (e: Int)
        ≪Fjern e fra t≫
    end Delete

    Proc Member[t: T] (e: Int) → (Bool)
        return ≪Indeholder t e?≫
    end Member

    Proc DeleteMin[t: T] → (Int)
        return ≪Det minimale tal i t≫
    end DeleteMin
end SP
```

b) Beskriv hvordan et tableau kan bruges til at implementere box'en SP, så alle operationer (undtagen evt Init) får udførselstid $O(h)$, hvor h er højden af tableauet.

   (Bemærk, at vi hermed benytter et tableau til at implementere både en prioritetskø og et søgetræ)

c) Angiv hvordan operationen Delete kan implementeres.

Opgave 26

Betragt box'en Eq side 83 i [P&D].

Tilføj en procedure Print, der udskriver klasserne i ækvivalensrelationen. Hvad er dens udførelsestid?

Opgave 27

Antag nu, at vi for N navngivne byer ønsker at opretholde information om, om de er forbundet med en vej eller ej (vejen kan evt. gå igennem andre byer). Vi ønsker med andre ord at opretholde en ækvivalensrelation, hvor to byer er ækvivalente, hvis der findes en vej imellem dem. Byer der ikke er ækvivalente kan så gøres ækvivalente, ved at vi bygger nye veje.

Hvordan skulle ækvivalensrelationen i [P&D] modificeres, hvis vi ønsker at kunne "bygge veje" mellem to navngivne byer, og spørge om to byer er forbundet med en vej (er ækvivalente)?

Opgave 28

En matrix kan repræsenteres v.hj.a. følgende type

(*) **Type** Matrix = **List** (**List** (Int))

Hvis M er en $n \times n$ matrix og x er en vektor af længde n, kan produktet mellem M og x beregnes v.hj.a. følgende procedure

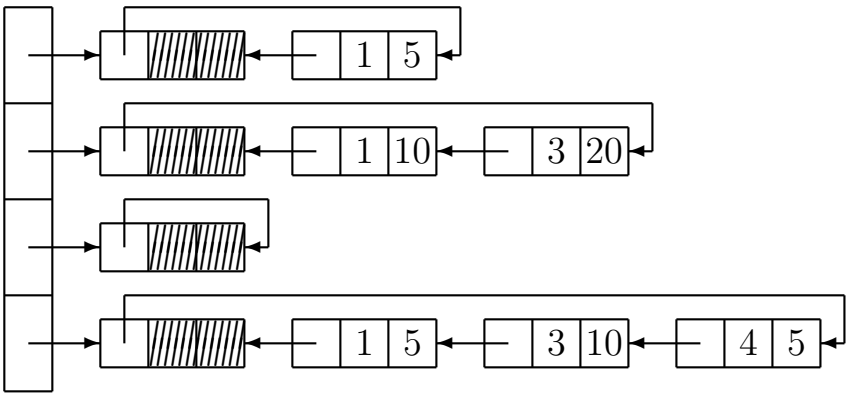
```
Proc Multiply[M: Matrix, x: Vector] → (Vector)
  (+ Var i: Int
     Var y: Vector
     i, y := 0, Vector(0||x|)
     do i < |x| →
           (+ Var j: Int
              j := 0
              do j < |x| →
                    y.(i) := y.(i)+M.(i, j)*x.(j)
                    j := j+1
              od
              i := i+1
           +)
     od
     return y
  +)
end Multiply
```

En matrix (og en vektor) siges at være *tynd*, såfremt hovedparten af dens elementer er 0. Store, tynde matricer og vektorer kan repræsenteres væsentligt mere hensigtsmæssigt ved i stedet for definitionen (*) at anvende en type, som gør det muligt kun at gemme de elementer, der er forskellige fra 0. Én af metoderne består i at repræsentere hver række v.hj.a. en kædet liste som illustreret ved følgende eksempel.

Matricen

$$M = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 10 & 5 \end{bmatrix}$$

repræsenteres på følgende måde



a) Definer passende TRINE-typer

Type Tvector = ≪Tynd vektor≫
Type Tmatrix = ≪Tynd matrix≫

som reflekterer denne repræsentation, og skitsér en TRINE-procedure

Proc Tmultiply[M: Tmatrix, x: Tvector] → (Tvector)

b) Antag nu, at der skal skrives en procedure, som multiplicerer to matricer. Med den sædvanlige typedefinition (*) ser denne ud som følger

```
Proc Product[M1, M2: Matrix] → (Matrix)
  (+ Var n, i: Int
     Var M: Matrix
     n := |M1|
     i, M := 0, List(List(0|n)|n)
     do i < n →
        (+ Var j: Int
           j := 0
           do j < n →
              (+ Var k: Int
                 k := 0
                 do k < n →
                    M.(i, j) := M.(i, j)+M1.(i, k)*M2.(k, j)
                    k := k+1
                 od
              +)
```

```
                    j:=j+1
                od
            +)
            i:=i+1
        od
        return M
    +)
end Product
```

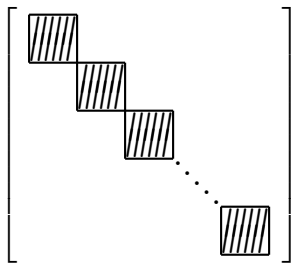
Forklar, hvorfor typen Tmatrix ikke er hensigtsmæssig med henblik på multiplikation af tynde *matricer*.

Angiv en mere hensigtsmæssig repræsentation og skitser, hvordan en effektiv multiplikationsprocedure

Proc Tproduct[Tm1,Tm2: Tmatrix] → (Tmatrix)

virker i dette tilfælde.

c) Antag at M1 og M2 er to (såkaldt *blokdiagonale*) $n \times n$ matricer af følgende form



hvor alle elementer uden for de skraverede blokke er 0; og hvor der er ialt $\sqrt{n}$ lige store blokke, som altså selv er $\sqrt{n} \times \sqrt{n}$ matricer.

Idet X er en vilkårlig vektor af længde n, skal størrelsesordenen af udførelsestiden for følgende procedurekald angives

i) Multiply[m,x]

ii) Tmultiply[tm,tx]

iii) Product[m1,m2]

iv) Tproduct[tm1,tm2]

hvor m1, m2 og x (tm1, tm2 og tx) er repræsentationer af M1, M2 og X.

Angiv også størrelsesordenen af det antal multiplikationer, der udføres i de fire procedurekald.

Opgave 29

a) Skitser en datastruktur Hestesko, hvis værdimængde er mængder af heltal, og hvor operationerne er:

$\text{Init}[\text{H})$ **sat** $\text{H}' = \emptyset$
$\text{Insert}[\text{H}](\text{i})$ **sat** $\text{H}' = \text{H} \cup \{i\}$
$\text{Member}[\text{H}](\text{i})$ **sat** $(\text{member}' = (\text{i} \in \text{H})) \wedge (\text{H}' = \text{H})$
$\text{DeleteMed}[\text{H,i}]$ **sat** $\text{H} \neq \emptyset \rightarrow$
$(\text{i'} \in \text{H}) \wedge (\text{H}' = \text{H} \setminus \{\text{i}'\}) \wedge$
$(|\{\text{h} \in \text{H} : \text{h} < \text{i}'\}| < \frac{|\text{H}|}{2} \leq |\{\text{h} \in \text{H} : \text{h} \leq \text{i}'\}|)$
$\text{Size}[\text{H}]$ **sat** $(\text{Size}' = |\text{H}| \wedge (\text{H}' = \text{H})$

DeleteMed fjerner *medianen* af elementerne i Hesteskoen, dvs. det element der ligger ca. i midten.

Operationerne skal have flg. tidskompleksiteter:

$\text{T}[\![\textbf{Init}[\text{H}]]\!] \in O(1)$
$\text{T}[\![\textbf{Insert}[\text{H}](\text{i})]\!] \in O(\log |\text{H}|)$
$\text{T}[\![\textbf{Member}[\text{H}](\text{i})]\!] \in O(\log |\text{H}|)$
$\text{T}[\![\textbf{DeleteMed}[\text{H}, \text{i}]]\!] \in O(\log |\text{H}|)$
$\text{T}[\![\textbf{Size}[\text{H}]]\!] \in O(1)$

b) Hvad nu hvis værdimængden er sække af heltal (jfr. opgave 22) i stedet for mængder af heltal?

Opgave 30

I [P&D] afsnit 9.2 er det vist, at en optælling af en binær tæller koster $O(1)$ amortiseret. Vi ønsker nu at lave følgende datatype:

Box BinærTæller
  Type Tal = **List**(Int)

  Proc Init [R: Tal]

```
        R := List(0 | n)
    end Init

    Proc Up[R: Tal]
        ≪R:=R+1≫
    end Up

    Proc Down[R: Tal]
        ≪R:=R-1≫
    end Down

    Proc Print[R: Tal]
        ≪Udskriv R≫
    end Print
end BinærTæller
```

Hvis vi (jfr. [P&D]) udfylder stumperne således:

```
where ≪R:=R+1≫ is
  i := 0
  do R.(i) = 1 → R.(i), i := 0, i+1 od
  R.(i) := 1

where ≪R:=R-1≫ is
  i := 0
  do R.(i) = 0 → R.(i), i := 1, i+1 od
  R.(i) := 0
```

opnår vi, at stumperne opfylder følgende specifikationer:

$$\ll\text{R:=R+1}\gg \textbf{ sat } f(R) < 2^n \rightarrow f(R') = f(R) + 1$$
$$\ll\text{R:=R-1}\gg \textbf{ sat } f(R) > 0 \rightarrow f(R') = f(R) - 1$$

f er her funktionen, så $f(R) = \sum_{j=0}^{n-1} R.(j) \cdot 2^j$.

Til gengæld kan vi nu ikke sige, at en sekvens bestående af k udførelser af Up og Down (i en vilkårlig rækkefølge) koster $O(k)$, idet der findes en

sekvens, hvor (næsten) hver udførelse af Up og Down koster $O(n)$. Dette gælder f.eks. hvis vi har tallet 01111111 og skiftevis bruger Up og Down.

Betragt i stedet følgende udfyldelse af stumperne, hvor vi tilføjer "cifferet" -1:

where ≪R:=R+1≫ **is**
  i := 0
  do { $f(R_{start})+1 = f(R)+2^i$ }
    R.(i) = 1 → R.(i), i := 0, i+1
  od
  if R.(i) = 0 → R.(i) := 1
  | R.(i) = -1 → R.(i) := 0
  fi
  { $f(R) = f(R_{start})+1$}

where ≪R:=R-1≫ **is**
  i := 0
  do { $f(R_{start})-1 = f(R)-2^i$ }
    R.(i) = -1 → R.(i), i := 0, i+1
  od
  if R.(i) = 0 → R.(i) := -1
  | R.(i) = 1 → R.(i) := 0
  fi
  { $f(R) = f(R_{start})-1$}

hvor vi stadig har $f(R) = \sum_{j=0}^{n-1} R.(j) \cdot 2^j$, og hvor R_{start} angiver R's værdi første gang invarianten mødes.

a) Giv et eksempel på, at repræsentationen af et tal ikke længere er entydig.

b) Vis, at programstumperne stadig opfylder specifikationerne.

c) Vis, at Up og Down nu har amortiseret udførelsestid $O(1)$.

Opgave 31

Gennemfør det udeladte argument i [P&D] kap. 9.4, dvs. vis at alle operationerne i box'en Sequence (pånær Print) har amortiseret udførselstid $O(1)$.

Opgave 32

Vis, hvorledes man kan realisere en kø v.hj.a. to stakke på en sådan måde, at den amortiserede udførelsestid for køens operationer tilhører $O(1)$.

Opgave 33

a) Betragt proceduren

Proc Pip[n: Int]
    skip
end Pip

Hvilke af følgende specifikationer opfylder den:

- Pip[n] **sat** true $\rightarrow$ false
- Pip[n] **sat** true $\rightarrow$ true
- Pip[n] **sat** false $\rightarrow$ true
- Pip[n] **sat** false $\rightarrow$ false
- Pip[n] **sat** true $\rightarrow \varphi(n')$
- Pip[n] **sat** false $\rightarrow \varphi(n')$
- Pip[n] **sat** $\varphi(n) \rightarrow \varphi(n')$
- Pip[n] **sat** $\varphi(n) \rightarrow$ true
- Pip[n] **sat** $\varphi(n) \rightarrow$ false

hvor φ er en vilkårlig betingelse.

b) Besvar spørgsmål a) i det tilfælde, hvor proceduren har udseendet

Proc Pip[n: Int]
    do true $\rightarrow$ **skip od**
end Pip

Opgave 34

Appendix A indeholder 4 sider fra Kingston: "Algorithms and Data Structures" og beskriver datastrukturen *Binomialkø*. Læs afsnittet og besvar følgende.

a) Vis, at $|B_r| = 2^r$ og højde$(B_r) = r + 1$.

b) Angiv et passende udvalg af procedurer i en box af formen

Box BQ
$\vdots$
end BQ

som understøttes effektivt af en binomialkø.

c) Find en simpel potentialfunktion for en binomialkø, som viser at de amortiserede udførelsestider for Insert og DelMin tilhører hhv. $O(1)$ og $O(\log(n))$.

Opgave 35

Et polynomium i en variabel er som bekendt et udtryk af formen

$$p(x) = a_n x^n + \ldots + a_1 x + a_0$$

Summen af $p(x)$ og polynomiet

$$q(x) = b_n x^n + \ldots + b_1 x + b_0$$

er polynomiet

$$(p + q)(x) = (a_n + b_n)x^n + \ldots + (a_1 + b_1)x + (a_0 + b_0)$$

og deres produkt er polynomiet

$$(p * q)(x) = c_{2n} x^{2n} + \ldots + c_1 x + c_0$$

hvor

$$c_m = \sum_{i=0}^{m} a_i * b_{m-i} \quad \text{for } 0 \leq m \leq 2n.$$

Betragt følgende box

Box PolyNom
  Type Poly = ≪Polynomium≫
  Proc Konst[p: Poly](k)
  Proc One[p: Poly]
  Proc Eval[p: Poly](x) → (Int)
  Proc Add[p, q, r: Poly]
  Proc Mult[p, q, r: Poly]
end

Angiv en hensigtsmæssig realisering af typen ≪Polynomium≫ og af de fem procedurer, når de skal opfylde følgende specifikationer

Konst[p](k) **sat** $p'(x) = k$
One[p] **sat** $p'(x) = x$
Eval[p](a) **sat** Eval$' = p(a)$
Add[p, q, r] **sat** $r = p + q$
Mult[p, q, r] **sat** $r = p * q$

og når det vides, at polynomierne er "tynde", dvs. at mange af koefficienterne er nul.

Opgave 36

I [G&AP] kap. 3.1 er der argumenteret for, at med den givne repræsentation har alle operationerne i den monotone mængde (H) undtagen DeleteSome udførelsestid $O(1)$. Der er også argumenteret for, at DeleteSome har amortiseret udførelsestid $O(1)$. Beskriv en repræsentation, der giver *alle* operationerne i den monotone mængde udførelsestid $O(1)$.

Opgave 37

a) Skriv en algoritme, der undersøger om en given ikke-orienteret graf (repræsenteret v.hj.a. kantlister) er et træ (et træ er en sammenhængende graf uden kredse). Angiv algoritmens udførelsestid.

b) En ikke-orienteret graf $G = (V, E)$ er som bekendt to-delt, hvis der findes et snit (V_1, V_2) i G, så alle kanter krydser dette snit. Skriv en algoritme, der afgør om en ikke-orienteret graf G er todelt, og som i givet fald også angiver et snit, der krydses af alle kanterne.

Opgave 38

Algoritme Graffarvning kan også formuleres rekursivt. Skriv en procedure

Proc Farv[G: Graph]

som farver G v.hj.a. en rekursiv procedure

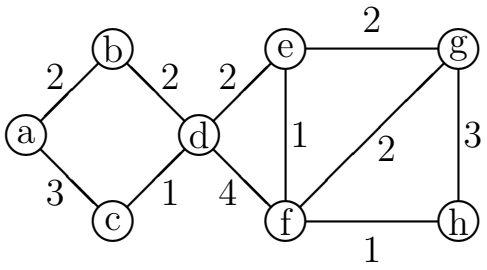
Proc Farvknude[G: Graph, C: Colours](i:int)

hvor Graph er en kantlisterepræsentation af G, C er en liste, der for hver knude angiver, om den er rød eller hvid og i er nummeret på den knude vi vil farve.

Angiv procedurens udførelsestid.

Opgave 39

Betragt følgende graf



a) Find et dybde-først udspændende træ, der starter ved a og ved d.

b) Find et bredde-først udspændende træ, der starter ved a og ved d.

c) Find et letteste udspændende træ ved hjælp af Kruskals og Prims algoritmer.

Opgave 40

Betragt følgende modifikation af letteste udspændende træ problemet: Der er givet en vægtet, sammenhængdende ikke-orienteret graf $G = (V, E)$ samt et antal kanter $e_1, \ldots, e_k \in E$. Skriv en algoritme, der finder ud af,

om der eksisterer et udspændende træ for G, der indeholder $e_1, \ldots, e_k$; og som i givet fald finder det letteste sådanne træ.

Opgave 41

Angiv hvorledes bredde-først søgning kan anvendes til at finde længderne af de korteste veje fra en given knude til de andre knuder i en graf (orienteret eller ikke-orienteret). Længden af en vej defineres i denne sammenhæng som antallet af kanter på vejen.

Opgave 42

Den *transitive* og *refleksive lukning* af en orienteret graf $G = (V, E)$ er grafen $G^t = (V, E^t)$, hvor $(v, w) \in E^t$, hvis $v = w$ eller der findes en vej i G fra v til w (bemærk at vi her tillader en kant at forbinde en knude med sig selv).

a) Angiv G_0^t, hvor G_0 er grafen side 3 i [G&AP].

b) Betragt incidensmatricen for G_0 side 5 i [G&AP]. Angiv $G_0 * G_0$, hvor $*$ er *Boolsk matrixmultiplikation*, der defineres som følger. Hvis A og B er $n * n$ Boolske matricer, er

$$A * B = C$$

hvor

$$C_{ij} = \bigvee_{k=0}^{n-1} a_{i,k} \wedge B_{k,j}$$

c) Observer, at det (i, j)'te element i $G_0 * G_0$ har værdien t hvis og kun hvis der findes en vej af længde præcis 2, der går fra i til j.

d) Vis, at der for enhver orienteret graf med incidensmatrix G_0 gælder, at incidensmatricen for dens transitive og refleksive lukning er givet ved

$$G_0^t = I \vee G_0 \vee G_0^2 \vee \ldots \vee G_0^{n-1} = (I \vee G_0)^{n-1}$$

hvor I er enhedsmatricen dvs.

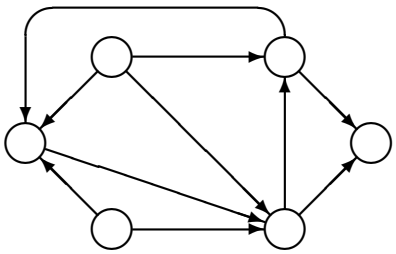
$$I = \begin{bmatrix} t & & & & \\ & t & & & \\ & & . & & \\ & & & . & \\ & & & & . \\ & & & & t \end{bmatrix}$$

e) Angiv en algoritme, der givet G_0 finder G_0^t. Angiv udførelsestiden.

Opgave 43

a) Skriv et TRINE program, der afgør om en orienteret graf er cyklisk, og som i givet fald finder en af grafens cykler.

b) Angiv hvorledes programmet virker på følgende graf



Opgave 44

I denne opgave betragtes træer, hvor knuderne har et variabelt antal sønner, og i hvilke alle knuder har forskellige mærkninger.

En *preorder* hhv. *postorder* udskrift af et sådant træ er resultatet af en udførelse af algoritmen "Tofarvning af træ" afsnit 4.1 i [G&AP], hvor

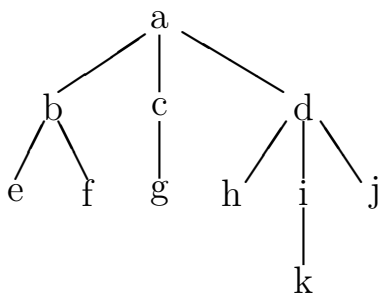
hvor

≪forfarv v≫ **is** ≪udskriv v's mærkning≫
≪efterfarv v≫ **is skip**

henholdsvis

≪forfarv v≫ **is skip**
≪efterfarv v≫ **is** ≪udskriv v's mærkning≫

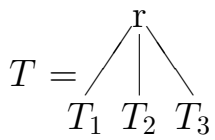
a) Angiv pre- og postorder udskriften af følgende træ



b) Vis, ved at angive modeksempler, at man ikke entydigt kan bestemme et træ ud fra enten dets preorder udskrift eller dets postorder udskrift.

c) Vis, at hvis man både har en preorder og en postorder udskrift af et træ, så kan man entydigt bestemme træet.

(Vink: Hvis



Så har vi:

$\text{Pre}(T) = \text{r Pre}(T_1)\ \text{Pre}(T_2)\ \ldots\ \text{Pre}(T_k)$

$\text{Post}(T) = \text{Post}(T_1)\ \text{Post}(T_2)\ \ldots\ \text{Post}(T_k)\ \text{r}$

Betragt første tegn i $\text{Pre}(T_1)$. Hvor står det i $\text{Post}(T)$?)

d) Skitser et program, der indlæser en preorder og en postorder udskrift af et træ og derefter konstruerer træet. Antag, at mærkningerne er ASCII-tegn.

Opgave 45

Betragt følgende algoritme

Algoritme: Tofarvning af graf
Stimulans : $G = (V, E)$, hvid graf

```
Respons  : G, med alle knuder farvet
Metode   : Proc ToFarv [G: Graf] (v: Int)
             ≪forfarv v ≫
             for w : (v, w) ∈ E do
               if ≪ w er ufarvet≫ → ToFarv[G](w) fi
             od
             ≪efterfarv v ≫
           end ToFarv

           for v ∈ V do
               if ≪ v er ufarvet≫ → ToFarv[G](v) fi
           od
```

Antag, at G er acyklisk. Hvilken sammenhæng er der mellem en efternummerering af G's knuder og en topologisk sortering af G?

Opgave 46

Hvorledes kan man løse følgende *tilrettelæggelsesproblem*: Der er givet et antal opgaver $T_1, \ldots, T_n$ (det kan være enkeltopgaver i forbindelse med f.eks. et byggeri), som har udførelsestider $t_1, \ldots, t_n$. Herudover er der angivet et antal begrænsninger af formen "T_i skal være færdig før T_j" (T_i kan f.eks. være "at rejse vægge" og T_j "at tjære tag"). Algoritmen skal angive, hvornår de enkelte opgaver kan startes, og den skal finde den minimale tid, der kræves for at færdiggøre alle opgaver.

Opgave 47

Den *vægtede* transitive og refleksive lukning af en vægtet orienteret graf G_w har samme kanter som den i opgave 42 definerede graf G_w^t, og vægten af en kant (v, w) i G_w^t er vægten af den letteste vej fra v til w i G_w (vægten af en vej er summen af vægtene af vejens kanter).

Den vægtede transitive lukning kan findes på (mindst) følgende måder

a) Dijkstras algoritme udført n gange.

b) Floyds algoritme.

c) Ved en modifikation af metoden i opgave 42, hvor $\wedge$ erstattes af $+$ og $\vee$ af minimum.

Gør i detaljer rede for hver af disse metoder og angiv deres udførelsestid for forskellige tæthedsgrader af grafen.

Opgave 48

Der findes en del numeriske algoritmer i hvilke der indgår beregning af midtpunktet af et interval $[a, b]$, hvor $a \leq b$.

a) Vis ved et eksempel at den matematiske identitet

$$(a + b)/2 = a + (b - a)/2$$

ikke gælder for flydende komma tal med endelig nøjagtighed.
(Vink: Find a og b så $(a \oplus b) \oslash 2 < a$)

b) Skriv en algoritme, som bruger *bisection* til at finde et nulpunkt for en kontinuert funktion $f(x)$ i et interval $[a, b]$, hvor det vides at $f(a) \cdot f(b) < 0$. Bisection foregår ved successive halveringer af intervallet $[a, b]$ på en måde, der minder om binær søgning. Dvs. først beregnes intervallets midtpunkt og derefter fortsættes der med det af de to "halve" intervaller, hvor funktionen vides at skifte fortegn.

Opgave 49

Skriv en algoritme, der som stimulans tager tre vilkårlige reelle tal, a, b og c og som respons angiver, om 2. gradsligningen $ax^2 + bx + c = 0$ har reelle løsninger, og i så fald hvilke.

Antag, at denne algoritme afvikles på en maskine, som har parametre

$$(\beta, s, m, M) = (10, 4, -99, 99)$$

og som approximerer v.hj.a. afskæring.

a) Udfør algoritmen med stimulans

$$a = 1, b = -2.136, c = 0.002857$$

b) Udregn de eksakte løsninger til ligningen $x^2 - 2.136x + 0.002857 = 0$ og sammenlign disse med dem, der blev fundet under a). Er værdierne fra a) tilfredsstillende?

c) Såfremt værdierne fra a) ikke er tilfredsstillende, hvad er årsagen? Kan løsningerne til en 2. gradsligning findes på en måde, der er mere hensigtsmæssig, numerisk set?

Opgave 50

Til løsning af følgende "iklædte opgave"

> En stige af højde h (mm) stilles lodret op ad en væg, hvorefter dens fod trækkes d (mm) ud fra væggen ad et vandret underlag. Hvor mange mm er stigens top kommet ned i forhold til før?

kan man overveje at bruge formlen

$$f(h,d) = h - \sqrt{h^2 - d^2}$$

a) Er det en god idé?

b) Hvis ikke, hvad kan man så gøre?

Opgave 51

Beregn værdierne $U_0, U_1, \ldots, U_n, \ldots$, defineret ved $U_{n+1} = (x+1)U_n - 1$, startende med $U_0 = 1/x$, for $x = 2, 3, 4, \ldots$. Forklar de observerede resultater.

Opgave 52

Betragt følgende integral

$$I_n = \int_0^1 x^n e^{(x-1)} dx$$

a) Vis, at

$$\begin{aligned} I_0 &= 1 - e^{-1} \\ I_n &= 1 - n * I_{n-1} \text{ for } n \geq 1 \end{aligned}$$

b) Vis, at

$$I_n - I_{n-1} = \int_0^1 (x^n - x^{n-1}) e^{(x-1)} dx < 0$$

og argumenter for at $I_n \to 0$ for $n \to \infty$.

c) Skriv et program der udregner $I_0, I_1, \ldots, I_n$ v.hj.a. formelen i a). Konvergerer følgen mod 0?

d) En alternativ formel til udregning af integralerne er

$$I_{k-1} = (1 - I_k)/k$$

hvor vi kan starte iterationen med at sætte $I_n = 0$ for en passende stor værdi af n og derefter benytte formlen for $i \leq k \leq n$. Virker det bedre?

Opgave 53

I [G&AP] er det vist, hvordan man kan multiplicere to n-cifrede heltal i en tid i $O(n^{\log_2 3})$. En oplagt idé er at forsøge at dele tallene op i flere dele som f.eks.

$$x = \boxed{u \mid v \mid w}$$
$$y = \boxed{r \mid s \mid t}$$

hvor hver del nu indeholder $p = n/3$ cifre.

a) Hvad skal k være i rekursionsligningen

$$T(n) \approx k * T(n/3) + n$$

hvis resultatet skal være bedre end $O(n^{\log_2 3})$?

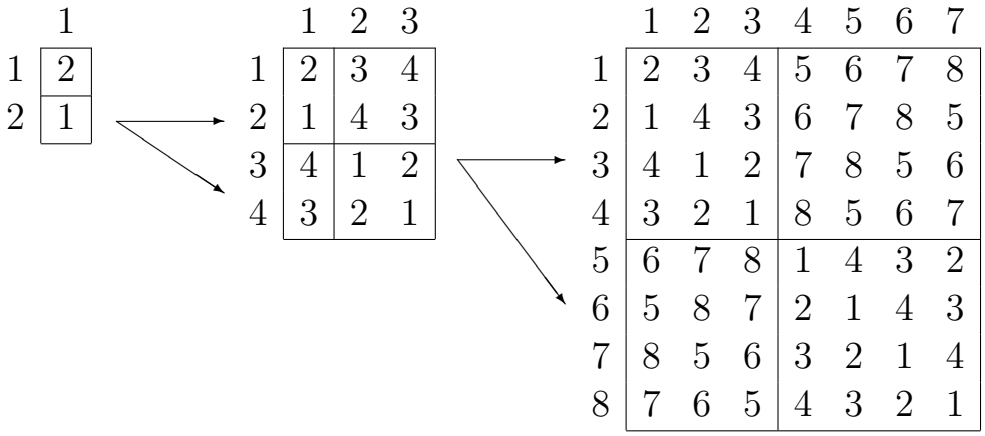
b) Hvad er det bedste k, du kan opnå? Hvad bliver udførelsestiden?

Opgave 54

Følgende tegning illustrerer hvordan man kan anvende del-og-kombiner teknikken til at konstruere en tennisturnering, hvor alle spiller mod alle. I eksemplet er der tale om 8 spillere, som hver spiller én kamp om dagen. Turneringen varer således 7 dage, og den i'te række i tabellen angiver den rækkefølge, i hvilken spiller nr. i møder de andre.

Gør rede for, hvordan denne konstruktion kan opfattes som en realisering af del-og-kombiner skabelonen.

Skriv et TRINE program, som for $k \geq 1$ konstruerer og udskriver en spilleplan for en turnering med 2^k spillere.



Opgave 55

Givet to polynomier

$$\begin{aligned} p(x) &= a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \\ q(x) &= b_m x^m + b_{m-1}x^{m-1} + \ldots + b_1 + b_0 \end{aligned}$$

Deres produkt er følgende polynomium af grad $n + m$

$$r(x) = p(x) * q(x) = c_{n+m}x^{n+m} + c_{n+m-1}x^{n+m-1} + \ldots + c_1 + c_0$$

hvor $c_i = \sum_{j+k=i} a_j * b_k$ for $0 \leq i \leq m + n$.

a) Vis, at den oplagte måde at udregne $r(x)$ har udførelsestid i $O((n + 1) \cdot (m + 1))$.

b) Antag, at $n = m$ og vis, at $r(x)$ kan udregnes i tid $O(n^{\log_2 3})$. (Vink: Et heltal som f.eks. 37916 kan opfattes som "polynomiet" $3 \cdot 10^4 + 7 \cdot 10^3 + 9 \cdot 10^2 + 1 \cdot 10 + 6$. Der er derfor en oplagt analogi mellem multiplikation af heltal og multiplikation af polynomier.)

Et polynomium kan repræsenteres på andre måder end ved sine koefficienter. Én sådan anden måde er ved sine rødder, hvor $n+1$ tal $a, r_1, r_2, \ldots, r_n$ nu repræsenterer polynomiet

$$R(x) = a * (x - r_1) * (x - r_2) * \ldots * (x - r_n).$$

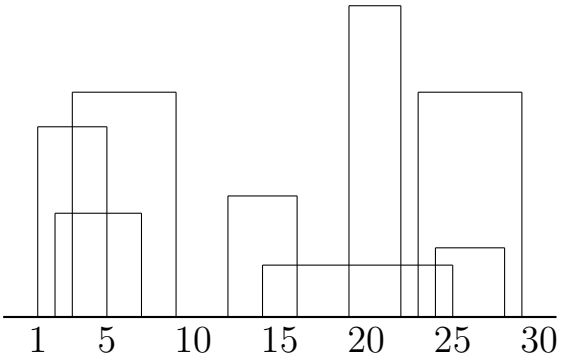
c) Konstruer en del-og-kombiner algoritme, der givet $a, r_1, \ldots, r_n$ beregner koefficienterne i $R(x)$. Hvad er algoritmens udførelsestid? (Hvis algoritmen udføres med stimuli $2, 2, 3$, skal den returnere $2, -10, 12$ fordi

$$R(x) = 2 * (x - 2) * (x - 3) = 2x^2 - 10x + 12$$

Opgave 56

Følgende tegning (der består af et antal rektangler) kan opfattes som repræsenterende bygninger i en by.

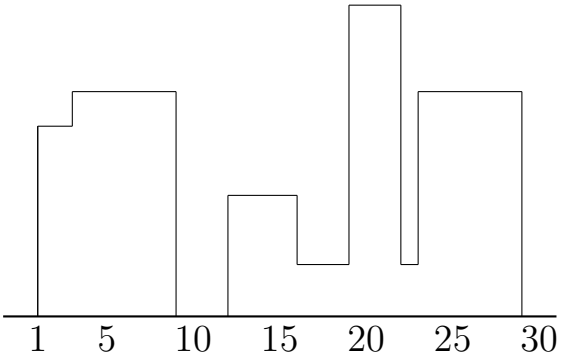
(*)



(a)

Byens *silhuet* repræsenteres af følgende tegning

(*)



(b)

Opgaven går ud på at skrive en del-og-kombiner algoritme, der læser en følge af elementer af formen (l_i, h_i, r_i), som hver angiver et rektangel, hvis venstre (højre) side har x-koordinat l_i (r_i), og hvis højde er h_i. "Byen" (*) repræsenteres således af følgende data

(1,11,5) (2,6,7) (3,13,9) (12,7,16) (14,3,25) (19,18,22) (23,13,29) (24,4,28)

Algoritmen skal producere en liste af tal af formen

(***) $(x_0, h_1, x_1, h_2, \ldots, x_{i-1}, h_i, x_i, \ldots, h_n, x_n)$

hvor x'erne er voksende og h_i angiver silhuethøjden mellem x_{i-1} og x_i. (**) repræsenteres således af følgende liste

(1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29)

a) Antag, at der er givet en silhuet af formen (***) samt en bygning (l, h, r). Hvordan opdateres silhuetten til også at omfatte denne bygning?

b) Udvid svaret på a) til at vise hvordan man kombinerer to silhuetter.

c) Skriv del-og-kombiner algoritmen for hele problemet og angiv dens udførelsestid.

Opgave 57

Vi betragter problemet Hanois Tårne fra [G&AP]. Som set ved forelæsningen kræves der mindst $2^n - 1$ enkeltflytninger for at flytte de n skiver. Argumentet er følgende induktionsargument, hvor induktionsantagelsen er

$H(n)$: der kræves $2^n - 1$ enkeltflytninger for at flytte n skiver.

Basis

Det er klart, at $H(1) = 1 = 2^1 - 1$.

Induktionsskridt

For at flytte n skiver skal vi først flytte de $n-1$ øverste over på hjælpestangen. Dette kræver iflg. induktionsantagelsen $2^{n-1} - 1$ flytninger.

Vi flytter nu den største skive til sin destination. Dette kræver 1 flytning.

Til slut flytter vi de $n - 1$ skiver over på den største. Dette kræver igen pr. induktionsantagelse $2^{n-1} - 1$ flytninger.

Altså kræves der til flytning af n skiver

$$(2^{n-1} - 1) + 1 + (2^{n-1} - 1) = 2^n - 1$$

dvs. $H(n)$ er bevist.

Det er nemt at skrive en rekursiv procedure

Proc Hanoi(n: Int, a, b, c: Stang)

der løser problemet for n skiver.

a) Argumenter for, at der højst foretages $6n$ *forskellige* procedurekald i denne løsning.

b) Betyder det ikke, at vi med dynamisk programmering kan løse problemet i tid $O(n)$?

Ovenfor har vi lige vist, at problemet kræver tid $\Omega(2^n)$.

c) Forklar dette tilsyneladende paradoks.

Opgave 58

I dette spørgsmål betragtes matrix produkter mellem ikke-kvadratiske matricer. Hvis A er en $m \times k$ matrix og B er en $k \times n$ matrix, så er deres produkt C en $m \times n$ matrix af formen

$$\boxed{A = \{a_{i,j}\}} \quad * \quad \boxed{B = \{b_{i,j}\}} \quad = \quad \boxed{C = \{c_{i,j}\}}$$

hvor

$$c_{i,j} = \sum_{p=1}^{k} a_{i,p} * b_{p,j}$$

Det følger af denne formel, at udregningen af matricen C involverer $m \cdot k \cdot n$ multiplikationer, hvilket betyder, at hvis mere end to matricer skal multipliceres, så er det ikke lige meget, hvordan man organiserer udregningen. Betragt som eksempel produktet

$$C = \underset{[1,100]}{A} \quad * \quad \underset{[100,2]}{B} \quad * \quad \underset{[2,1000]}{C}$$

hvor matricerne på højresiden har de angivne dimensioner.

Hvis dette produkt udregnes på følgende måde

$$A * (B * C)$$

skal der udføres

$$100 \cdot 2 \cdot 1000 + 1 \cdot 100 \cdot 1000 = 300000$$

multiplikationer, hvorimod følgende udregning

$$(A * B) * C$$

kun anvender

$$1 \cdot 100 \cdot 2 + 1 \cdot 2 \cdot 1000 = 2200$$

multiplikationer.

Dimensionerne af tre sådanne matricer kan angives v.hj.a. følgende sekvens af heltal

(1,100,2,1000)

og i almindelighed kan dimensionerne i et produkt mellem n matricer angives v.hj.a. en sekvens af formen

(*) $(m_0, m_1, \ldots, m_i, \ldots, m_n)$

Hvis vi med $c(i, j)$ betegner det minimale antal multiplikationer i udregningen af "produktet"

$$(m_i, m_{i+1}, \ldots, m_j)$$

hvor $i < j$, så er det nemt at se, at

$$c(i,j) = \begin{cases} 0 & \text{hvis } i+1 = j \\ min_{i<k<j} & \{c(i,k) + c(k,j) + m_i \cdot m_k \cdot m_j\} \end{cases}$$

Skriv et TRINE program, der givet en følge af formen (*) beregner $c(0, n)$. Angiv programmets tidskompleksitet.

Opgave 59

I forbindelse med kontrol af at ord er korrekt stavet, er det bekvemt at have et mål for *afstanden* mellem to ord. Ét sådant mål defineres i det følgende.

Givet to ord, s og t, definerer vi deres *maksimale fælles delord*, mfd(s, t), som det længste ord u, for hvilket der gælder, at u kan fremkomme fra *både* s og t ved at slette nul eller flere tegn. F.eks. haves

mfd(abekat,abba) = aba

At maksimale fælles delord ikke nødvendigvis er entydige ses af

$$\text{mfd(cykel,cykle)} = \begin{cases} \text{cyke} \\ \text{cykl} \end{cases}$$

Længden af det maksimale fælles delord er derimod entydig, og vi kan derfor definere *Svejgaard Koefficienten* ved

$$Sv(s,t) = 2 * \frac{|\text{mfd}(s,t)|}{|s| + |t|}$$

(idet vi underforstår, at $Sv(\lambda, \lambda) = 1$).

Bemærk, at $Sv(s,s) = 1$ og $Sv(s,t) = 0$ hvis s og t ikke har fælles tegn.

Skriv en TRINE procedure

Proc lmfd(s, t: Text) → (Int)

der finder $|\text{mfd}(s,t)|$ og brug den i en procedure

Proc $Sv(s, t$: Text) → (Q'Tal)

der beregner Svejgaard Koefficienten.
(Typen Q'Tal stammer fra opgave U50.)

Angiv udførelsestiden for procedurerne.

Opgave 60

Proceduren Tbin og algoritmen Pascals Trekant i [G&AP] bruger lige så megen plads som de bruger tid, i begge tilfælde $\Theta(n * m)$.

Kan du finde en forbedring til Pascals Trekant, så den kun bruger *lineær* plads, dvs. plads $O(n)$?

Referencer

[**Bentley**] J. Bentley: Programming Pearls. Addison-Wesley Publishing Company, 1986.

[**Dat 1–nr. 3**] Opgavesamling Datalogi 1 (dProg1).

[**G&AP**] E.M. Schmidt: Grafalgoritmer og Algoritmisk Problemløsningsteknik. Datalogisk Afdeling, Aarhus Universitet, 1995.

[**P&D**] E.M. Schmidt & M.I. Schwartzbach: Programmeringsteori og Datastrukturer. DAIMI FN-56, Datalogisk Afdeling, Aarhus Universitet, januar 1995.

[**TRINE**] E.M. Schmidt & M.I. Schwartzbach: Programmering og programmeringssproget Trine. DAIMI FN-36, Datalogisk Afdeling, Aarhus Universitet, august 1994.