Opgave 1

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

b) Find for hver af algoritmerne en formel, der i lighed med tabellen side 75 i [Bentley], udtrykker algoritmens udførelsestid i millisekunder. For at få nogle realistiske formler er det vigtigt at bruge nogle store datasæt på nogle millioner tal. For at generere sådanne datasæt kan bruges følgende metode, der generere et tilfældigt heltal mellem i og j.

```
public static int random(int i, int j) {
        return ( (int) Math.floor(i + (j-i)*Math.random()) );
}
```

Til måling af tiden kan metoden `System.currentTimeMillis()` bruges.

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

Opgave 2

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 3

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

Algoritme heltalsdivision(a, b):
Input: heltal a og b, hvor $a \geq 0, b > 0$
Output: heltal q og r, hvor $a = q * b + r$ og $0 \leq r < b$

b) I følgende algoritme repræsenterer array'et c koefficienterne i polynomiet

$$p_c(x) = x^n + c[n-1]x^{n-1} + \ldots + c[1]x + c[0]$$

Skriv følgende algoritme, og bevis at den er korrekt.

Algoritme polynomium(c, a):
Input: Koefficienterne c i $p_c(x)$ og en værdi a
Output: Værdien $r = p_c(a)$

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

Algoritme multiplikation(m, n):
Input: Heltal m og n, hvor $m, n \geq 0$
Output: Produktet $r = m * n$

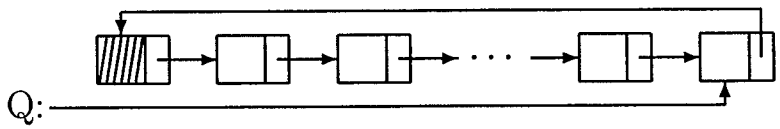
Opgave 4

Betragt et array X af heltal som i maksimal delsumsproblemet. Beskriv en algoritme der finder det delarray, der har sum tættest på nul.

Hint: Lad H være et "hjælpearray", så $H[i] = \sum_{j=0}^{i} X[j]$ og betragt tallene i H sorteret.

Opgave 5

Skriv i JAVA en klasse, der implementerer ADT Queue, side 61 i [G&T] v.hj.a. en kædet liste af følgende form



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

Opgave 6

Find tidskompleksiteten af Exa, Exb og Exc udtrykt ved n:

```
class Reinforcable {
  int[][] B,C;
  int[] A;

  public void Exa(int n) {
    int i,j,k;

    for (i=0; i<n; i++) {
      for (j=0; j<n; j++) {
        C[i][j]=0;
        for (k=0; k<n; k++) {
          C[i][j]=C[i][j]+A[i]*B[k][j];
        }
      }
    }
  }

  public void Exb(int n) {
    int i,j,k;

    for (i=1; i<n; i=2*i) {
      for (j=0; j<i; j++) {
        k=2;
        while (k<n) k = k*k;
      }
    }
  }

  public void Exc(int n) {
    int i,j;

    for (i=1; i<=n; i++) {
      if (i%2==1) {
        j=i;
        while (j<=n) j++;
        j=1;
        while (j<=i) j++;
      }
    }
  }
}
```

Opgave 7

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.

a) Skriv følgende algoritme.

Algoritme periode(f):
Input: periodisk funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$
Output: heltallet p, hvor p er f's periode

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

Opgave 8

I denne opgave betragtes arrays, hvis elementer er *cifre*, det vil sige arrays af formen

$$X = (1, 2, 5, 3, 7, 8, 2, 0, 5, 8, 9, 6, 8, 9, 8)$$

Et *monotont afsnit* i et sådant array er et delarray hvori værdierne er ikke aftagende; fx er $X[3..5] = (3, 7, 8)$ et monotont afsnit i X, mens $X[1..3] = (2, 5, 3)$ ikke er et monotont afsnit. Vi betegner med $\text{llma}(X)$ længden af det længste monotone afsnit i X, og med $\text{lsma}(X)$ længden af det sidste monotone afsnit i X. For ovenstående X er $\text{llma}(X) = 4$ og $\text{lsma}(X) = 1$.

a) Argumenter for at følgende algoritme er korrekt (find blandt andet en passende invariant og bevis, at den er gyldig).

Algoritme længsteMonotoneAfsnit(X):
Input: Array X med længde n
Output: længden af længste monotone afsnit $r = \text{llma}(X)$.

$r \leftarrow 1$; $h \leftarrow 1$; $i \leftarrow 1$
while $i \neq |X|$ **do** $\{I\}$
  if $X[i-1] \leq X[i]$ **then**
    $h \leftarrow h + 1$
  else
    $h \leftarrow 1$
  if $h > r$ **then**
    $r \leftarrow h$
  $i \leftarrow i + 1$

I stedet for et monotont afsnit i et array kan man interessere sig for en *monoton følge* i array'et, hvor en følge adskiller sig fra et afsnit ved, at dens elementer ikke behøver "af hænge sammen". Følgen

$$(2, 3, 5, 6, 8, 9)$$

udgør således den med understregning angivne monotone følge i

$$X = (1, \underline{2}, 5, \underline{3}, 7, 8, 2, 0, \underline{5}, 8, 9, \underline{6}, \underline{8}, \underline{9}, 8)$$

b) Konkretiser de udkommenterede ubestemte stumper, så følgende algoritme bliver gyldig og korrekt. Med $\text{llmf}(X)$ betegner vi længden af den længste monotone følge i X, og med $\text{llmf}_d(X)$ betegnes længden af den længste monotone følge i X, hvis sidste ciffer er D.

Algoritme længsteMonotoneFølge(X):
Input: Array X med længde n
Output: længden r af længste monotone følge $\text{llmf}(X)$.

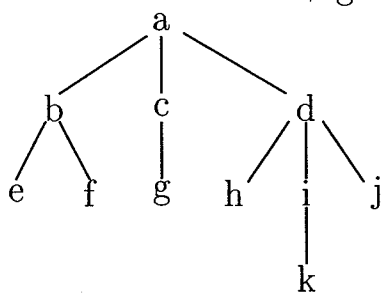
// Initialiser i og h
while $i \neq |X|$ **do** $\{(h[d] = \text{llmf}_d(X[0..i]) \text{ for } 0 \leq d \leq 9) \wedge (0 \leq i \leq |X|)\}$
    // Iterer
// Beregn r

Opgave 9

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, hvor knudens mærkning udskrives under besøget ("visit action") ved preorder hhv. postorder gennemløbet af træet jvf. afsnit 2.3.3 i [G&T].

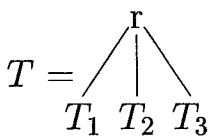
a) Angiv pre- og postorder udskriften af følgende træ



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 10

Følgende er en abstrakt klasse for repræsentation af matricer i JAVA;

```
abstract class Matrix {
  public int rows, columns;

  // get returnerer indholdet af indgangen i række i og søjle j
  abstract public int get(int i,int j);

  // set sætter indholdet af indgangen i række i og søjle j til v
  abstract public void set(int i, int j, int v);

  // multiply returnerer matrix-produktet af denne matrix og søjlvektoren .
  abstract public Matrix multiply(Matrix x) throws RuntimeException;
}
```

Objekter af klassen Matrix kan repræsenteres ved hjælp af arrays, og en implementation kunne se ud som følger:

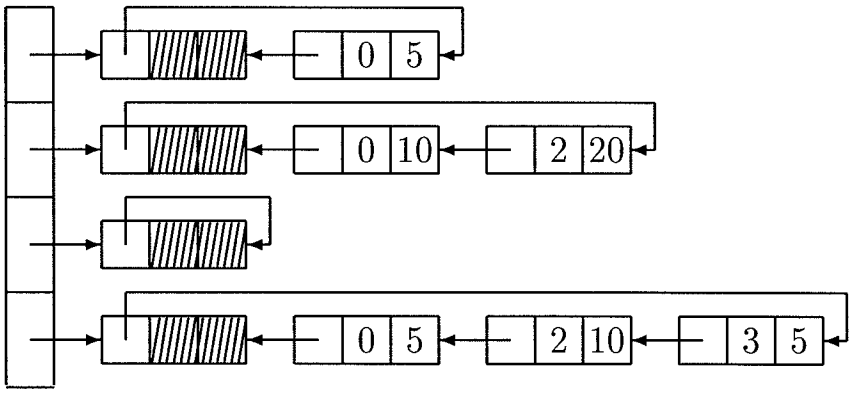
```
class ArrayMatrix extends Matrix {
  private int[][] M;

  ArrayMatrix(int r, int c) {
    rows=r;
    columns=c;
    M = new int[r][c];
  }

  public int get(int i, int j) {
    return M[i][j];
  }

  public void set(int i, int j, int v) {
    M[i][j]=v;
  }

  public Matrix multiply(Matrix x) throws RuntimeException {
    ArrayMatrix result;
    if (x.rows != columns || x.columns!=1)
      throw new RuntimeException("Incompatible dimensions for multiply.");
    else {
      result = new ArrayMatrix(1,rows);
      for (int i=0; i<rows; i++) {
        result.set(i,0,0);
        for (int j=0; j<rows; j++)
          result.set(i,0, result.get(i,0) + get(i,j)*x.get(j,0));
      }
    }
    return result;
  }
}
```

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 ArrayMatric 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) Beskriv en klasse TyndMatrix, der implementerer Matrix og skitser en algoritme for metoden `multiply`.

b) Antag nu, at klassen Matrix udvides med en metode, som multiplicerer to matricer:

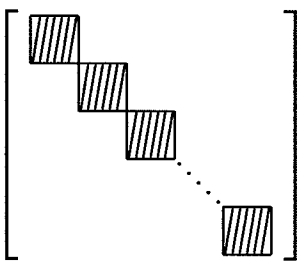
```
// product returnerer matrix-produktet af denne matrix og m
abstract public Matrix product(Matrix m) throws RuntimeException;
```

For ArrayMatrix kan en implementationen for `product` se ud som følger:

```
public Matrix product(Matrix m) throws RuntimeException {
  ArrayMatrix result;
  if (m.rows != columns)
    throw new RuntimeException("Incompatible dimensions for product.");
  else {
    result = new ArrayMatrix(rows, m.columns);
    for (int i=0; i<rows; i++) {
      for (int j=0; j<m.columns; j++) {
        result.set(i,j,0);
        for (int k=0; k<columns; k++)
          result.set(i,j, result.get(i,j) + get(i,k)*m.get(k,j));
      }
    }
  }
  return result;
}
```

Forklar hvorfor den nuværende repræsentation for TyndMatrix er uhensigtmæssig med henblik på multiplikation af tynde matricer. Angiv en mere hensisgtmæssig repræsentation og skitser, hvordan en effektiv implementation for `product` for TyndMatrix virker.

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 metodekald angives.

i) `m1.product(m2)` i ArrayMatrix

ii) `m1.multiply(x)` i TyndMatrix

iii) `m1.multiply(x)` i ArrayMatrix

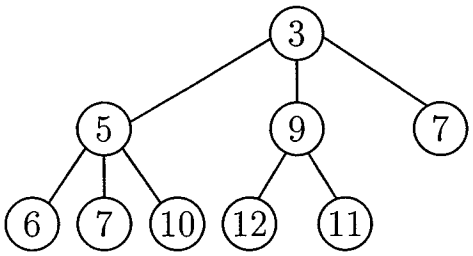
iv) `m1.product(m2)` i TyndMatrix

hvor `m1`, `m2` og `x` er repræsentationer af $M1$, $M2$ og X.

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

Opgave 11

En heap er iflg. [G&T] afsnit 2.4.3 et perfekt balanceret *binært* træ, hvor det nederste lag er fyldt så meget som muligt fra venstre mod højre. En d-heap 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 heap som beskrevet i [G&T] er således en 2-heap. Følgende er et eksempel på en 3-heap.



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

b) Antag, at en d-heap 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 12

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 entydigt bestemt af sin karakteristiske funktion,

$$\chi_M : G \to \{\textsf{true}, \textsf{false}\},$$

hvor

$$\chi_M(e) = \textsf{true} \Leftrightarrow e \in M.$$

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

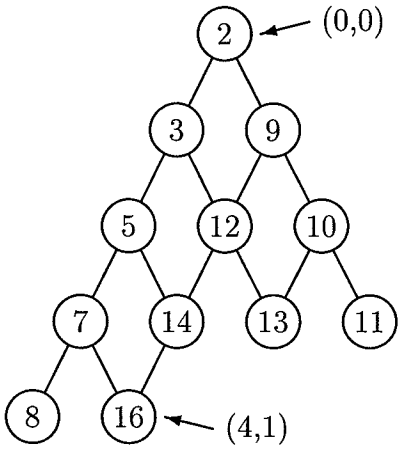
$$S = S_1 + S_2 \Leftrightarrow \forall e \in G : \chi_S(e) = \chi_{S_1}(e) + \chi_{S_2}(e)$$

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 13

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?

b) Betragt følgende abstrakte datastruktur:

```
public interface SP {
  // Query methods
  public boolean member(int e);
  // Update methods
  public void insert(int e);
  public void delete(int e);
  public int deleteMin();
}
```

Beskriv hvordan et tableau kan bruges til at implementere den abstrakte datastruktur SP, så alle operationer får udførselstid $O(h)$, hvor h er højden af tableauet. Beskrivelsen skal (naturligvis) omfatte en constructor for klassen.

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

Opgave 14

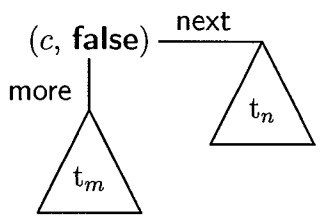
En *trie* er en datastruktur til opbevaring af *ord*, hvor man søger at genbruge fælles præfikser. I Java kan vi definere

```
public class trie {
  private char letter;
  private boolean word;
  private trie next, more;
  ⋮
}
```

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.

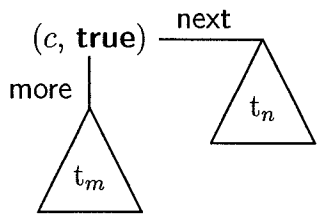
Formelt kan vi definere $words(\mathrm{t})$, der er mængden af ord i en trie t:

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



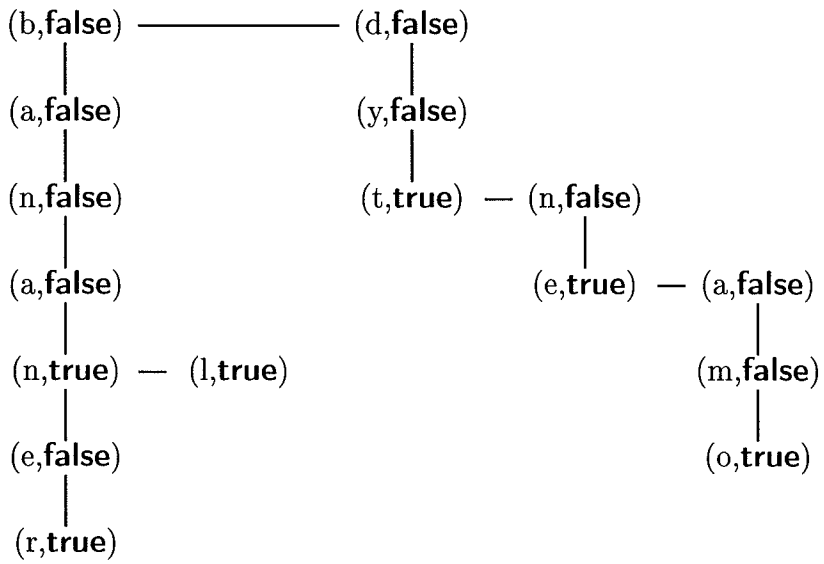
  så er $words(\mathrm{t}) = \{cw \mid w \in words(\mathrm{t}_m)\} \cup words(\mathrm{t}_n)$

- hvis t er af formen



  så er $words(\mathrm{t}) = \{c\} \cup \{cw \mid w \in words(\mathrm{t}_m)\} \cup words(\mathrm{t}_n)$

Hvis t fx er følgende trie



så er

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

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

b) Lav en trie-baseret implementation af følgende abstrakte datastruktur:

```
public interface Vocabulary {
  // Query methods
  public boolean member(String s);
  public void print();
  // Update methods
  public void insert(String s);
}
```

hvor metoderne har følgende effekter:

new trie() skaber en ny trie, t, med $words(\mathrm{t}) = \emptyset$.

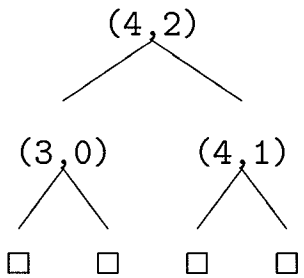
t.member(w**)** returnerer **true** hvis ordet w er i trien t og ellers **false**.

t.print() Udskriver $words(\mathrm{t})$ tilsvarende ovenfor.

t.insert(w**)** indsætter ordet w i trien t, dvs. hvis t er trien før operationen, og t′ er trien efter operationen, så er $words(\mathrm{t}') = words(\mathrm{t}) \cup \{\mathrm{w}\}$.

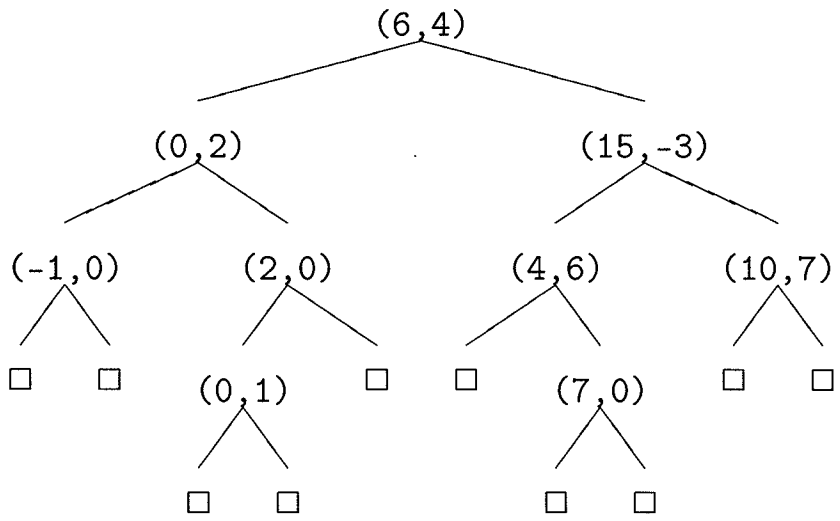
Opgave 15

I denne opgave ser vi på en variant af *søgetræer*, der indeholder heltal. Den sædvanlige implementation er modificeret således, at hver knude foruden en værdi indeholder en *addend*, der implicit er lagt til alle værdier i det undertræ, som den er rod i. Indholdet af en knude angives som et par (*værdi*, *addend*). For eksempel indeholder træet



elementerne 5, 6 og 7.

a) Hvilke elementer indeholder søgetræet



b) Angiv, hvordan de følgende operationer kan realiseres, hvor vores nye datastruktur kaldes VTree. vt er en variabel af type VTree

- **new VTree()** i tid $O(1)$

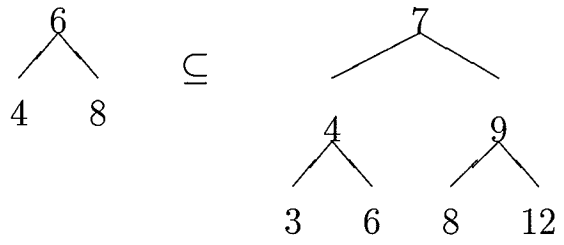
- **vt.member**(i) og **vt.insert**(i) i tid $O(h)$
- **vt.plus**(k) i tid $O(1)$. Denne operation lægger tallet k til alle elementer i vt.
- **vt.skub**(i, k) i tid $O(h)$. Denne operation lægger tallet $k \geq 0$ til alle de elementer i vt, der er $\geq i$.

Her angiver h højden af det aktuelle søgetræ.

c) Vis, hvordan man kan sikre, at højden af søgetræerne altid er logaritmisk i antallet af elementer i søgetræet. Dette kan man fx gøre ved at vise hvordan den rød-sorte strategi kan tilpasses denne variant.

Opgave 16

Med søgetræer kan man repræsentere mængder af elementer. Givet to søgetræer T_1 og T_2 kan man spørge om $T_1 \subseteq T_2$, det vil sige om mængden af elementer i T_1 er indeholdt i mængden af elementer i T_2. For eksempel gælder inklusionen



I det følgende antager vi, at træerne er *balancerede*, at T_1 har n knuder, og at T_2 har m knuder. Vi skal afgøre $T_1 \subseteq T_2$ under forskellige krav til tid og plads.

a) Hvordan kan man afgøre $T_1 \subseteq T_2$ i tid $O(n \log m)$

b) Argumenter for, at hvis $n \in \Theta(m)$, så er tid $O(n+m)$ bedre end tid $O(n \log m)$.

c) Hvordan kan man afgøre $T_1 \subseteq T_2$ i tid $O(n+m)$, hvis der må bruges $O(n+m)$ ekstra plads.

d) Kan du også afgøre $T_1 \subseteq T_2$ i tid $O(n+m)$, hvis der kun må bruges $O(\log n + \log m)$ ekstra plads?

Opgave 17

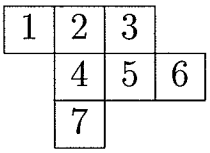
a) Formuler følgende spil som et transitionssystem:

Man har en krukke med sorte og hvide kugler, dobbelt så mange hvide som sorte. To tilfældige kugler tages op. Hvis de er ens bliver de smidt væk, og man lægger en ny sort kugle ned i krukken. Hvis de er forskellige, smider man den sorte væk og putter den hvide tilbage.

Det er klart at spillet terminerer (hvorfor?). Kan der siges noget om farven på den sidste kugle?

b) Formuler følgende spil som et transitionssystem.

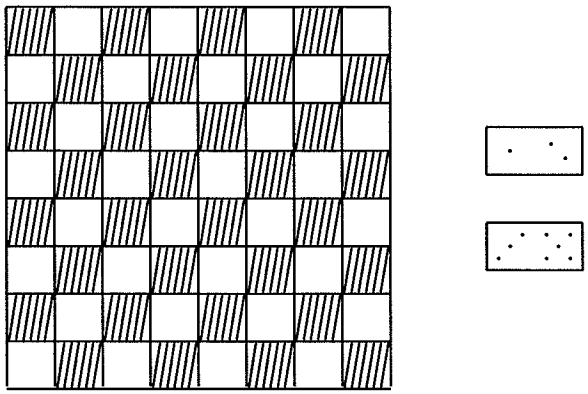
Et spillebræt har udseendet



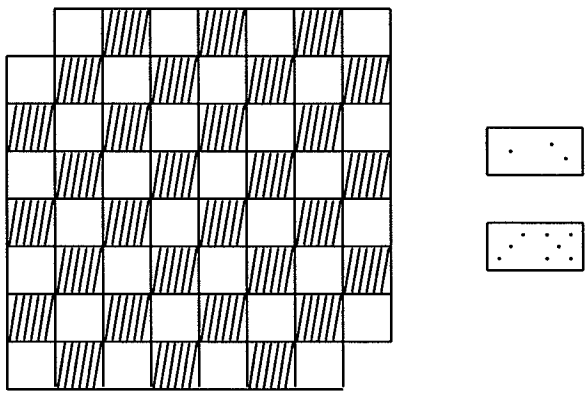
Fra et felt med et lige nummer kan man gå lodret eller vandret til et nabofelt. Fra et felt med et ulige nummer kan man gå diagonalt til et nabofelt.

Opgave 18

Det er klart, at et skakbræt kan dækkes fuldstændigt v.hj.a. 32 dominobrikker.



Hvordan stiller situationen sig, hvis man fjerner to hjørner og spørger om følgende bræt kan dækkes af 31 dominobrikker.



Formuler svaret som et udsagn om et passende transitionssystem.

Opgave 19

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

hestesko(): Returnerer den tomme hestesko.
insert(i): Indsætter elementet i i hesteskoen.
member(i): Svarer `true` hvis i ligger i hesteskoen.
`false` ellers.
deleteMed(): Hvis hesteskoen ikke er tom så returneres og slettes det element hvorom der gælder at mindre end halvdelen af hesteskoens elementer er mindre end det og højst halvdelen er større.
size(): Returnerer størrelsen af (antallet af elementer i) hesteskoen.

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

Operationerne skal have flg. tidskompleksiteter:

Operation	Ønsket tid
hestesko	$O(1)$
insert	$O(\log n)$
member	$O(\log n)$
deleteMed	$O(\log n)$
size	$O(1)$

b) Hvad nu hvis værdimængden er sække af heltal (jfr. opgave 12) i stedet for mængder af heltal?

Opgave 20

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 interface:

```
public interface PolyNom {
  // Initialiseringsfunktioner
  public void konst(int k);
  public void one();
  // Operationer på flere polynomier
  public int eval(x);
  public PolyNom add(PolyNom p, PolyNom q);
  public PolyNom mult(PolyNom p, PolyNom q);
}
```

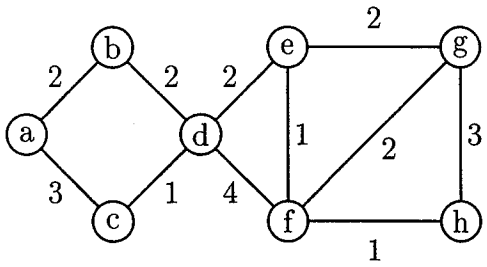
Angiv en hensigtsmæssig realisering af interfacet, når det skal opfylde følgende specifikationer:

konst(k)	Sætter polynomiet til $p(x) = k$
one()	Sætter polynomiet til $p(x) = x$
eval(a)	Evaluerer polynomiet i punktet a og returnerer $p(a)$.
add(p, q)	Returnerer $(p+q)(x)$.
mult(p, q)	Returnerer $(p * q)(x)$.

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

Opgave 21

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 22

a) Skriv en algoritme, der undersøger om en given ikke-orienteret graf repræsenteret v.hj.a. nabolister (adjacency lists jvf. 6.2.2 i [G&T]) er et træ (et træ er en sammenhængende graf uden cykler). Angiv algoritmens udførelsestid.

b) En ikke-orienteret graf $G = (V, E)$ er to-delt, hvis der findes en disjunkt opdeling af knuderne V i V_1 og V_2, så alle kanter i grafen G forbinder en knude fra V_1 med en knude fra V_2. Skriv en algoritme, der afgør om en ikke-orienteret graf G er todelt, og som i givet fald også angiver opdelingen i V_1 og V_2.

Opgave 23

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) En incidensmatrix for en graf med n knuder er en $n * n$ Boolsk matrix hvis (i, j)-te indgang er true (false) hvis der (ikke) findes en kant fra knude i til knude j. Angiv incidentmatricen G_0 for grafen i figur 6.17 a) i [G&T] .

b) 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.

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

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

Opgave 24

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 25

Den *vægtede* transitive og refleksive lukning af en vægtet orienteret graf G_w har samme kanter som den i opgave 23 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) Floyd og Warshalls algoritme.

c) Ved en modifikation af metoden i opgave 23, 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 26

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 27

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 28

I [G&T] afsnit 12.2.2 er det vist, hvordan man kan multiplicere to n-cifrede heltal i tid $O(n^{\log_2 3})$. 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

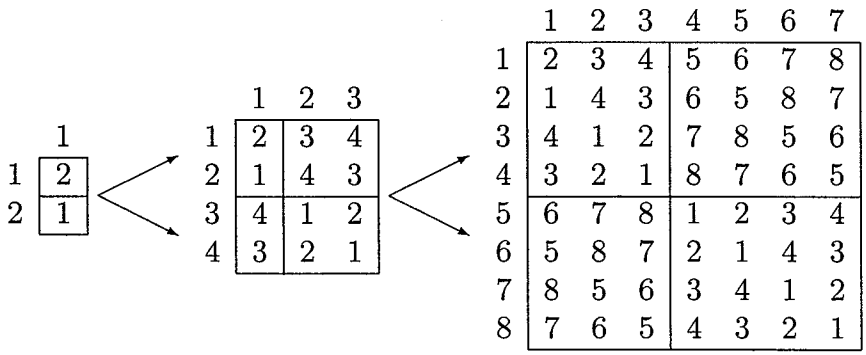
$$T(n) \approx k \cdot T(n/3) + n$$

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

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

Opgave 29

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.



	1	2
1	2	1
2	1	2

	1	2	3
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

a) Angiv ud fra tegningen, hvordan problemet "lav en tennisturnering for n spillere" kan løses vha. del-og-kombiner teknikken, dvs. angiv, hvordan et stort problem opdeles i del-problemer, hvordan små problemer umiddelbart løses, samt hvordan løsninger til del-problemer kombineres til en løsning til hele problemet. Du kan antage at n er en potens af 2.

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

Opgave 30

Antag at G er en ikke-orienteret vægtet graf med n knuder og m kanter, hvor vægtene er heltal mellem 0 og n.

a) Vis, hvordan man kan konstruere et letteste udspændende træ for G i tid

$$O(m \cdot F(n) + n \cdot U(n))$$

hvor $F(n)$ ($U(n)$) er den amortiserede udførelsestid for **find** (**union**) i en implementation af **Partition** (jfr. 12.1.4).

b) Hvor "gode" kan $F(n)$ og $U(n)$ blive?

Opgave 31

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 32

Vi betragter problemet Hanois Tårne. 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 flytnin g.

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

void Hanoi(**int** n, Stang a, Stang b, Stang c)

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 $O(n)$?

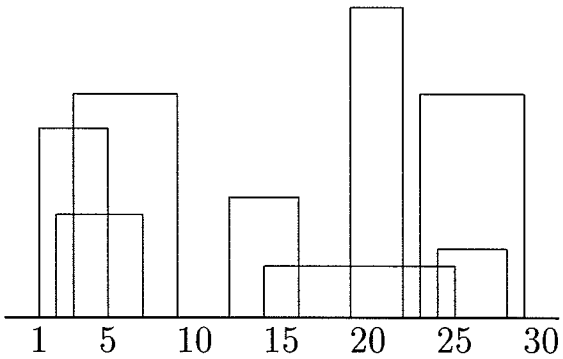
Ovenfor har vi lige vist, at problemet kræver tid $\Omega(2^n)$.

c) Forklar dette tilsyneladende paradoks.

Opgave 33

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

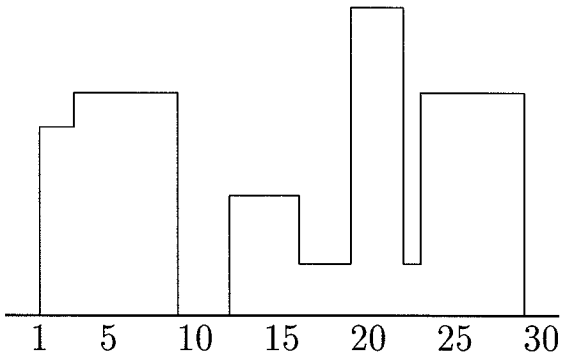
(*)



(*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.

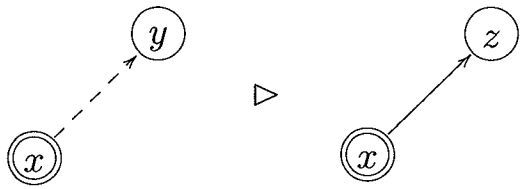
c) Skriv del-og-kombiner algoritmen for hele problemet og angiv dens udførelsestid.

Opgave 34

I denne opgave betragtes algoritmer der farver og mærker orienterede acykliske grafer. Mærkerne er heltal, der skrives inde i knuderne. En knude uden indgående kanter kaldes en rod.

Betragt følgende transitionssystem.

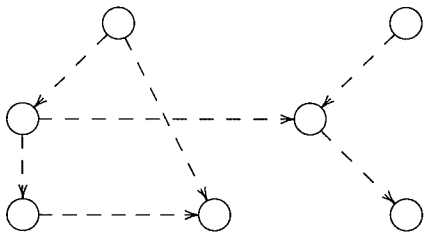
hvis den hvide indgrad er 0



hvor $z = \max\{x + 1, y\}$

Det er klart, at enhver proces for transitionssystemer er endelig, og det kan også vises (men det forlanges ikke), at enhver proces hvis startkonfiguration er en hvid orienteret acyklisk graf, ender i en konfiguration, hvor grafen er helt rød.

a) Angiv slutkonfigurationen for en proces, hvis startkonfigurationen er følgende graf



b) Hvad "gør" transitionssystemet i almindelighed, dvs. hvad angiver mærkerne i en slutkonfiguration. Bevis påstanden v.h.a.

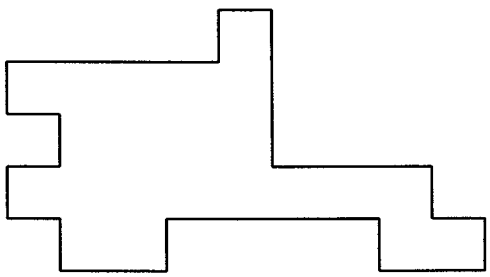
et passende invariansargument. (Vink: indfør f.eks. betegnelsen "en lyserød vej" for en vej i grafen, hvor alle kanter og knuder, pånær evt. den sidste knude, er røde.)

c) Gør rede for, hvordan man kan skrive en algoritme, der "realiserer" transitionssystemet. Hvad bliver udførelsestiden.

Opgave 35

To spillere, A og B, skiftes til at sætte hhv. en rød og en blå streg af længde 1 på et stykke kvadreret papir. Hver streg må placeres vilkårligt, vandret eller lodret, men selvfølgelig ikke hvor der er en streg i forvejen.

A begynder (og tegner med rødt), og hans opgave er at lave en sammenhængende lukket "kurve", som fx



der udelukkende består af røde streger. B's opgave er at forhindre A i at lave en sådan kurve.

En af spillerne har en vindende strategi. Hvem?

Vink: Opstil en passende invariant for spilleplanens udseende efter hvert B-træk.

Opgave 36 Potensopløftning

Betragt følgende algoritme:

Algoritme: Lineær potensopløftning(x, p)
Inputbetingelse : $p \geq 0$
Outputkrav : $r = x^p$
Metode : $r \leftarrow 1;$
$q \leftarrow p;$
$\{I\}$while $q > 0$ do
$r \leftarrow r * x;$
$q \leftarrow q - 1$

– hvor I er udsagnet $(rx^q = x^p) \wedge (q \geq 0)$.

a) Angiv de bevisbyrder (på formen sekvens-af-tilordninger), der fremkommer i et gyldighedsbevis for algoritmen.

b) Gennemfør bevisbyrderne.

c) Angiv en termineringsfunktion.

d) Konkludér, at algoritmen korrekt.

Opgave 37 Heltalskvadratod

Heltalskvadratroden af et tal $n \geq 0$ er det tal $r \geq 0$, der opfylder

$$r^2 \leq n < (r+1)^2.$$

a) Argumentér for, at følgende algoritme er gyldig og korrekt:

Algoritme: Lineær heltalskvadratrod(n)
Inputbetingelse : $n \geq 0$
Outputkrav : $r^2 \leq n < (r+1)^2$
Metode : $a \leftarrow 0;$
$b \leftarrow n;$
$\{I\}$while $(n < b*b) \vee (n > a*(a+2))$ do
if $n < b*b$ then
$b \leftarrow b-1$
else
$a \leftarrow a+1;$
$r \leftarrow (a+b)/2$

– hvor I er udsagnet $(a^2 \leq n < (b+1)^2) \wedge (a, b \geq 0)$.

b) Ovenstående algoritme kan siges at beskrive en *lineær* søgning efter kvadratroden. Her følger en algoritme, der benytter *binær* søgning:

Algoritme: Binær heltalskvadratrod(n)
Inputbetingelse : $n \geq 0$
Outputkrav : $r^2 \leq n < (r+1)^2$
Metode : $a \leftarrow 0;$
$b \leftarrow n+1;$
$m \leftarrow (n+1)/2;$
$\{I\}$while $(n < m*m) \vee (n \geq (m+1)*(m+1))$ do
if $n < m*m$ then
$b \leftarrow m;$
$m \leftarrow (a+b)/2$
else
$a \leftarrow m+1;$
$m \leftarrow (a+b)/2;$
$r \leftarrow m$

– hvor I er udsagnet $(a^2 \leq n < b^2) \wedge (m = (a+b)/2)$.

Argumentér som i a) for, at algoritmen er gyldig og korrekt.

c) Bevis, at følgende er endnu en korrekt måde at beregne heltalskvadratroden på:

Algoritme: Heltalskvadratrod(n)
Inputbetingelse : $n \geq 0$
Outputkrav : $r^2 \leq n < (r+1)^2$
Metode : $r \leftarrow 0;$
$s \leftarrow 1;$
$\{I\}$while $s \leq n$ do
$r \leftarrow r+1;$
$s \leftarrow s+2*r+1$

– hvor I er udsagnet $(s = (r+1)^2) \wedge (r^2 \leq n)$.

d) Sammenlign de tre algoritmer. Hvilken er "bedst"? Hvorfor?

Opgave 38 Lange heltal

Denne opgave drejer sig om at håndtere ikke-negative heltal af vilkårlig længde. Et sådant heltal repræsenteres ved hjælp af en liste på følgende måde: Tallet 29355891081 repræsenteres som $[1, 8, 0, \ldots, 3, 9, 2]$.

Det mindst betydende ciffer er altså den 0'te indgang i listen. Tallene repræsenteres *uden* foranstillede nuller, så tallet 0 repræsenteres med den tomme liste. Hvis X er en liste, vil vi skrive $\text{tal}(X)$ om det repræsenterede tal. Formelt er

$$\text{tal}(X) = \Sigma_{i=0}^{|X|-1} X[i] \cdot 10^i.$$

a) Skriv programstumpen S^{end}, så følgende algoritme bliver korrekt. Bevis korrekthed ved hjælp af en passende invariant, I. (Bemærk, at der undervejs i beregningen anvendes en repræsentation *med* foranstillede 0'er)

Algoritme: Sum(X, Y)
Inputbetingelse : X, Y lange heltal, $|X| \geq |Y|$
Outputkrav : $\text{tal}(Z) = \text{tal}(X) + \text{tal}(Y)$
Metode : $Z \leftarrow$ liste af længde $|X|$ med 0 i alle indgange;
$Y \leftarrow$ Y forlænget til længde $|X|$ med foranstillede 0'er;
$i \leftarrow 0$; $m \leftarrow 0$;
$\{I\}$while $i \neq |X|$ do
$Z[i] \leftarrow X[i] + Y[i] + m$;
if $Z[i] \leq 9$ then
$m \leftarrow 0$
else
$Z[i] \leftarrow Z[i] - 10$; $m \leftarrow 1$;
$i \leftarrow i + 1$
S^{end}

b) Betragt nu følgende algoritme:

Algoritme: Produkt(X, Y)
Inputbetingelse : X, Y lange heltal
Outputkrav : $\text{tal}(Z) = \text{tal}(X) \cdot \text{tal}(Y)$
Metode : $i \leftarrow 0$;
S^{init};
$\{I\}$while $i \neq |Y|$ do
S^{loop};
$i \leftarrow i + 1$

– hvor I er udsagnet $(\text{tal}(Z) = \text{tal}(X) \cdot \text{tal}(Y[0..i])) \wedge (0 \leq i \leq |Y|)$.

Bevis, at hvis denne algoritme er gyldig, så er den også korrekt. Skriv S^{init} og S^{loop}, så algoritmen bliver gyldig. (Man kan bruge Sum til dette.)

c) Angiv udførelsestiderne for algoritmerne under a) og b). Kan du foreslå en repræsentation af lange heltal, der er specielt velegnet, hvis de fleste af cifrene er 0? Begrund svaret.

Opgave 39 Variationer over Euklid

Betragt algoritmen Udvidet Euklid fra afsnit 6.4.1 i [H&S].

a) Vis, at algoritmen også er korrekt såfremt løkkens krop erstattes af

$$
\begin{array}{l}
\textsf{if } p > q \textsf{ then} \\
\qquad S^{\text{then}}; \\
\qquad p \leftarrow p - x * q; \quad s \leftarrow s + x * t \\
\textsf{else} \\
\qquad S^{\text{else}}; \\
\qquad q \leftarrow q - x * p; \quad t \leftarrow t + x * s
\end{array}
$$

– hvor x er en hjælpevariabel, og S^{then} og S^{else} ikke ændrer på p og q, men tilfredsstiller bevisbyrderne

$$\{0 < q < p\} S^{\text{then}} \{0 < x * q < p\} \qquad \{0 < p < q\} S^{\text{else}} \{0 < x * p < q\}.$$

b) En vigtig sætning i talteorien siger, at der for vilkårlige positive heltal m og n findes ikke-negative heltal a og b, således at $\text{sfd}(m, n) = am - bn$. Skriv en version af Euklids algoritme, der givet m og n beregner a og b. Følgende skitse til en algoritme kan være nyttig.

Algoritme: Euklid(m, n)
Inputbetingelse : $m, n \geq 0$
Outputkrav : $\text{sfd}(m, n) = am - bn$
Metode : S^{init};
$\{I\}$while $p \neq q$ do
if $p > q$ then
S^{then}
else
S^{else}

– hvor I er udsagnet

$$(\text{sfd}(p, q) = \text{sfd}(m, n)) \wedge (p = am - bn) \wedge (q = cn - dm) \wedge (p, q \geq 1) \wedge (a, b, c, d \geq 0).$$

Opgave 40 Maksimalt delprodukt

Lad A være en liste af reelle tal, i hvilken alle elementer er mindst 0. Det *maksimale delprodukt* af A, $\mathrm{mdp}(A)$ er defineret ved

$$\mathrm{mdp}(A) = \max_{0 \le j \le j' \le |A|} A[j] \cdot A[j+1] \cdot \cdots \cdot A[j'-1].$$

Dermed gælder for eksempel, at for

$$\begin{aligned} A &= [5.7, 2.3, 0.4, 8.2, 2.7, 0.999] \\ B &= [0.5, 0.2, 0.87] \end{aligned}$$

er $\mathrm{mdp}(A) = A[0] \cdot A[1] \cdot \cdots \cdot A[4] = 116.10216$ og $\mathrm{mdp}(B) = 1$. Bemærk, at et "tomt" produkt har værdien 1. Det følgende er en skitse af en algoritme, der beregner det maksimale delprodukt:

Algoritme: Maksimalt delprodukt(A)
Inputbetingelse : $A[0..n] \in \mathbf{R}_0$
Outputkrav : $r = \mathrm{mdp}(A)$
Metode : $i \leftarrow 0$;
S^{init};
$\{I\}$while $i \neq |A|$ do
S^{loop};
$i \leftarrow i+1$

– hvor I er udsagnet

$$\begin{aligned} & m = \mathrm{mdp}(A[0..i]) && \wedge \\ & h = \max_{0 \le j \le i} A[j] \cdot A[j+1] \cdot \cdots \cdot A[i-1] && \wedge \\ & 0 \le i \le |A|. \end{aligned}$$

Skriv S^{init} og S^{loop}, så algoritmen bliver gyldig og korrekt. Giv derefter et korrekthedsbevis.

Opgave 41 Fibonacci

Det n'te Fibonaccital, F_n, er defineret ved:

$$\begin{aligned} F_0 &= 1 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad \text{for } n > 1. \end{aligned}$$

a) Bevis, at følgende algoritme er gyldig og terminerer:

Algoritme: Fibonacci(n)
Inputbetingelse : $n \geq 0$
Outputkrav : $r = F_n$
Metode : $r \leftarrow 1; \quad s \leftarrow 1; \quad i \leftarrow 0;$
$\{I\}$while $i \neq n$ do
$r \leftarrow s;$
$s \leftarrow s + r;$
$i \leftarrow i + 1$

– hvor I er udsagnet $(r = F_i) \wedge (s = F_{i+1}) \wedge (0 \leq i \leq n)$.

b) Lad i det følgende R og K betegne 2×2-matricer med indgangene

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \qquad K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$

Produktet af R og K er defineret som

$$R \cdot K = \begin{bmatrix} r_{11}k_{11} + r_{12}k_{21} & r_{11}k_{12} + r_{12}k_{22} \\ r_{21}k_{11} + r_{22}k_{21} & r_{21}k_{12} + r_{22}k_{22} \end{bmatrix}$$

Vis, at for ethvert $n \geq 0$, vil beregningen

$$\begin{aligned} &K \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}; \\ &R \leftarrow K^n; \\ &r \leftarrow r_{11} \end{aligned}$$

sætte r til det n'te Fibonaccital. (Husk, at K^0 er identitetsmatricen, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.)

Hvordan ville du implementere denne algoritme?

Opgave 42 Metodologi

Brug programmeringsmetodologien fra [H&S] kapitel 7 til at skrive følgende algoritmer:

a) Givet en liste A af heltal og et heltal p, skal A opdeles i to dele, $A[0..r]$ og $A[r..|A|]$, således at tallene i den første (anden) del er højst (mindst) p:

Algoritme: Opdel(A, p)
Inputbetingelse : $A[0..n] \in \mathbf{Z}$
Outputkrav : $(A[0..r] \leq p) \wedge (A[r..\|A\|] \geq p)$
Metode : ?

b) Givet heltal $n \geq 1$ og $b \geq 2$, er vi interesserede i heltalsdelen af $\log_b n$, dvs. det heltal r, der opfylder $b^r \leq n < b^{r+1}$:

Algoritme: Heltalslogaritme(n, b)
Inputbetingelse : $n \geq 1, b \geq 2$
Outputkrav : $b^r \leq n < b^{r+1}$
Metode : ?

c) I en sorteret liste A af heltal vil vi kalde en sekvens af ens værdier for et *plateau*. Algoritmen skal finde længden af det længste plateau i A, $\mathrm{llp}(A)$:

Algoritme: Længste plateau(A)
Inputbetingelse : A sorteret, $\|A\| > 0$
Outputkrav : $r = \mathrm{llp}(A)$
Metode : ?

d) Et polynomium kan repræsenteres med en liste af koefficienter. F.eks. vil listen $A = [3, 1, 0, -8]$ repræsentere polynomiet

$$p_A(x) = 3 + x - 8x^3.$$

Algoritmen Værdi(A, x) skal beregne $p_A(x)$. Med A som ovenfor og $x = 1$ skal vi have resultatet $3 + 1 - 8 \cdot 1^3 = -4$.

Algoritme: Værdi(A, x)
Inputbetingelse : $A[0..n] \in \mathbf{Z}, x \in \mathbf{Z}$
Outputkrav : $r = p_A(x)$
Metode : ?

(NB: Algoritmen kan skrives, så den har udførelsestid $O(n)$.)