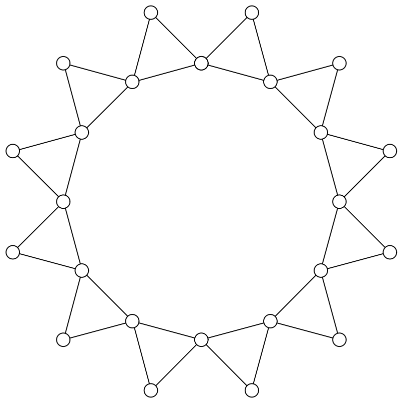
Skriftlig Eksamen
Algoritmer og Datastrukturer (dADS)

Datalogisk Institut
Aarhus Universitet

Mandag den 28. maj 2001, kl. 09–13

Opgave 1 (30%)

En *solsikke-graf* med omkreds k er en uorienteret graf bestående af k trekanter sat sammen i en kreds. For $k = 12$ ser grafen sådan ud:



Spørgsmål a: Angiv sammenhængen mellem antallet m af kanter og antallet n af knuder i en solsikke-graf. □

Vi lader nu kanterne i solsikke-grafen være vægtede med ikke-negative tal. Vi ønsker at finde dels korteste veje fra en given spids i grafen til alle andre knuder, dels et minimum udspændende træ for grafen.

Spørgsmål b: Argumentér for, at både Dijkstras algoritme for korteste veje og Kruskals algoritme for minimum udspændende træ bruger $\Omega(n \log n)$ tid på en solsikke-graf. □

Spørgsmål c: Beskriv en $O(n)$ algoritme, som givet en reference til en spids i grafen (d.v.s. en knude af grad to) finder længden af de korteste veje til alle andre knuder. Beskriv ligeledes en $O(n)$ algoritme, som finder et minimum udspændende træ for en vægtet solsikke-graf. Du skal argumentere for algoritmernes kompleksitet, og for at de finder det ønskede. Antag at grafen som sædvanligt er givet ved en *adjacency list* representation. □

Man kan uden bevis bruge, at der gælder følgende sætning om minimum udspændende træer:

> *Lad $G = (V, E)$ være en vægtet uorienteret graf, og lad $K \subseteq E$ være en kantmængde, som indeholder et minimum udspændende træ for G. Hvis K indeholder en cykel C og hvis e er en kant i C af størst vægt, da vil $K - \{e\}$ også indeholde et minimum udspændende træ for G.*

Opgave 2 (25%)

I forbindelse med fremstilling af tabeller for matematiske standardfunktioner (logaritmer, trigonometriske funktioner m.v.) har man traditionelt benyttet såkaldte *itererede differenser*. Ideen i denne metode er, at man først approksimerer den funktion, der skal tabellægges, med et polynomium, og dernæst udnytter, at man kan udregne værdien af et polynomium på *ækvidistante* punkter v.hj.a. en iterativ metode, der kun anvender additioner.

Følgende algoritme påstås at beregne værdierne af andengradspolynomiet

$$p(x) = ax^2 + bx + c$$

i punkterne $0, 1, 2, \ldots, n$.

Algoritme: ID(a, b, c, n)

Input: a, b, c: koefficienter i polynomiet $p(x)$
$n \geq 0$: tabelstørrelsen

Output: T: $T[i] = ai^2 + bi + c$ for $0 \leq i \leq n$

Metode: $T[0] \leftarrow c$; $t \leftarrow a + b$; $k \leftarrow 0$

$\{I\}$

while $k < n$ **do**

  $T[k+1] \leftarrow T[k] + t$;

  $t \leftarrow t + a + a$;

  $k \leftarrow k + 1$

Her er I udsagnet

$$(T[i] = ai^2 + bi + c \text{ for } 0 \leq i \leq k) \ \wedge \ (k \leq n) \ \wedge \ (t = a(2k+1) + b).$$

Spørgsmål a: Angiv hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. □

Spørgsmål b: Eftervis bevisbyrderne fra spørgsmål **a**. □

Spørgsmål c: Argumentér for, at algoritmen er korrekt. □

Opgave 3 (20%)

Lad $x = x_1x_2 \ldots x_n$, $y = y_1y_2 \ldots y_m$ og $z = z_1z_2 \ldots z_{n+m}$ være tre strenge af længde henholdsvis n, m og $n+m$. Vi kalder z et *flet* af x og y, hvis x og y findes som to disjunkte delsekvenser i z, og disse tilsammen udgør hele z.

> Eksempler: `gulerod` er et flet af `uro` og `gled`, og `dalgatorastritukturmerer` er et flet af `algoritmer` og `datastrukturer`.

For $0 \leq i \leq n$ og $0 \leq j \leq m$ lader vi $F[i,j]$ være en boolsk værdi, der angiver, hvorvidt strengen $z_1z_2 \ldots z_{i+j}$ er et flet af strengene $x_1x_2 \ldots x_i$ og $y_1y_2 \ldots y_j$. Her defineres $x_1x_2 \ldots x_i$ som den tomme streng, når $i = 0$ (og tilsvarende for y og z).

$F[i,j]$ kan beskrives ved følgende rekursionsformel:

$$F[i,j] = \begin{cases} X_{ij} \vee Y_{ij}\,, & i,j \geq 1 \\ X_{ij}\,, & i \geq 1, j = 0 \\ Y_{ij}\,, & i = 0, j \geq 1 \\ \texttt{Sand}\,, & i,j = 0, \end{cases}$$

hvor X_{ij} og Y_{ij} er udsagnene

$$\begin{aligned} X_{ij} &= (z_{i+j} = x_i \wedge F[i-1,j]), \\ Y_{ij} &= (z_{i+j} = y_j \wedge F[i,j-1]). \end{aligned}$$

Spørgsmål a: Opskriv tabellen for F, når x er `uro`, y er `gled` og z er `gulerod`. □

Spørgsmål b: Beskriv i form af pseudo-kode en algoritme baseret på dynamisk programmering, som i tid $O(nm)$ afgør, hvorvidt z er et flet af x og y. Der kræves et argument for, at algoritmen har den angivne kompleksitet. □

Spørgsmål c: Gør rede for, hvordan algoritmen kan udvides til i tilfælde af et positivt svar også at returnere indekserne for en delsekvens af z, som er lig x. □

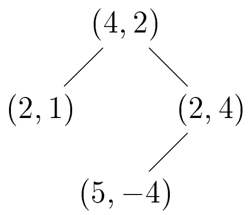
Opgave 4 (25%)

I denne opgave betragtes søgetræer, der indeholder heltal. Repræsentationen af tallene er imidlertid usædvanlig, idet hver knude indeholder et par,

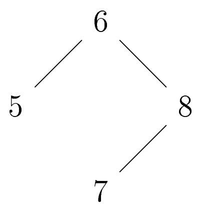
$$(basis, addend),$$

hvor *addend* skal opfattes som et tal, der skal lægges til samtlige baser i det undertræ, hvori knuden er rod.

Følgende træ

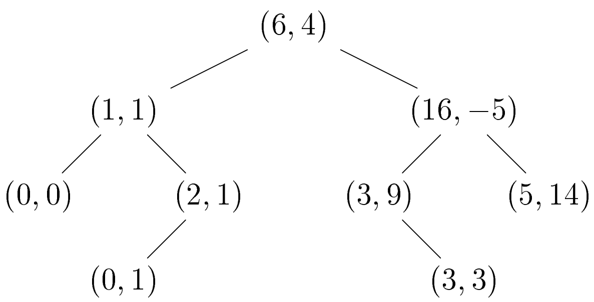


repræsenterer således det normale søgetræ



idet en knudes resulterende værdi opnås som summen af dens basis samt alle addender på vejen fra roden til knuden.

Spørgsmål a: Angiv det normale søgetræ, der repræsenteres af træet



□

Spørgsmål b: Angiv, hvordan følgende metoder kan realiseres:

$$\begin{array}{ll} \textsc{find}(k) & \text{i tid } O(h) \\ \textsc{insert}(k) & \text{i tid } O(h) \\ \textsc{plus}(d) & \text{i tid } O(1) \\ \textsc{shift}(k,d) & \text{i tid } O(h) \end{array}$$

hvor h er træets højde. Her er $\textsc{plus}(d)$ en operation, der lægger tallet d til alle værdier i træet, og $\textsc{shift}(k,d)$ er en operation, der lægger tallet $d \geq 0$ til alle værdier i træet, der er større end eller lig k. □

Spørgsmål c: Forklar, hvordan man kan sikre at h er $O(\log n)$, ved at angive hvordan de repræsenterede værdier kan bevares under rebalanceringsoperationer. Her er n antallet af elementer i træet. □