

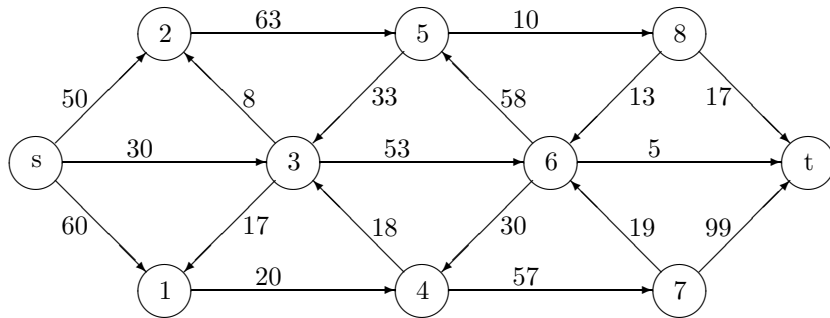
Algoritmik eksamen

sommeren 99

4 timers skriftlig prøve

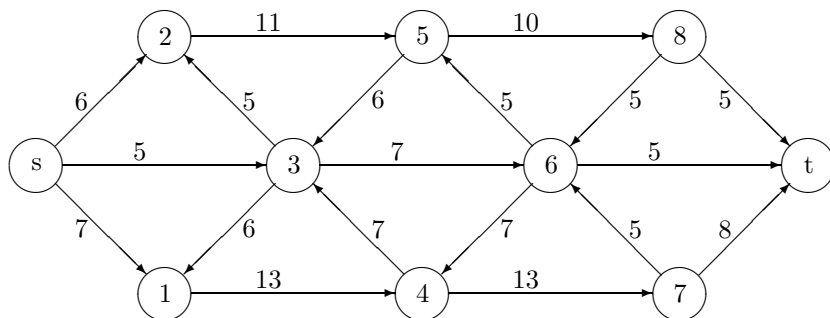
Opgave 1 (30%)

Betragt følgende strømningsnetværk (tallene på kanterne angiver kapaciteterne, c):



Strømningsnetværk N med kapaciteter (c) på kanterne.

Betragt følgende strømning f i N :



En strømning f i netværket N .

Spørgsmål 1A

Hvad er $f(\{s, 1, 2, 3, 4\}, \{5, 6, 7, 8, t\})$?

□

Spørgsmål 1B

Hvad er $c(\{s, 1, 2, 3, 4\}, \{5, 6, 7, 8, t\})$?

□

Spørgsmål 1C

Hvad er kapaciteten af et minimalt snit i N ?

□

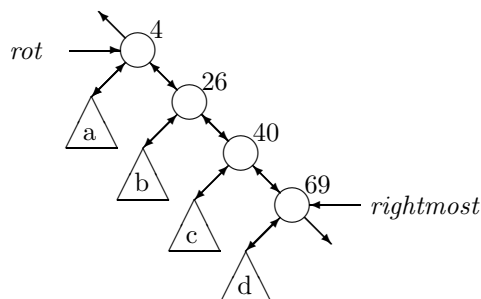
Opgave 2 (20%)

Binære træer, udvidet med to ekstra pointere, kan bruges til at implementere en datastruktur, der understøtter følgende operationer på mængder af elementer taget fra et univers med ordning:

- $init(S, e)$ Danner en struktur S , der kun indeholder e .
- $insert(S, e)$ Tilføjer e til strukturen S .
- $report(S, e)$ Rapporterer alle elementer i S , som er mindre end e (rækkefølgen af elementerne, der rapporteres, er ligegyldig).

Hver mængde implementeres ved et binært træ indeholdende elementer i heap orden. Dvs at et element i en knude er mindre end elementerne lagret i eventuelle efterfølgere. Hver knude indeholder udover et element tre pointere. En *up*-pointer, som peger på knudens forgænger. En *left*- og en *right*-pointer, der peger på henholdsvis venstre og højre efterfølger. De to ekstra pointere er *root*, som peger på roden og *rightmost*, som peger på det element man når ved at starte i roden og følge *right*-pointere så langt som muligt.

Eksempel:



Et binært træ T med tilhørende pointere.

I træet T ovenfor er elementerne i a større end 4 og ordnet i heap order indbyrdes, elementerne i b større end 26 og ordnet i heap order indbyrdes, o.s.v.

$init(S, e)$ implementeres ved at oprette en enkelt knude indeholdende e og lade de to pointere *root* og *rightmost* pege på denne knude. *up*-, *right*- og *left*-pointerne i knuden sættes til *nil*.

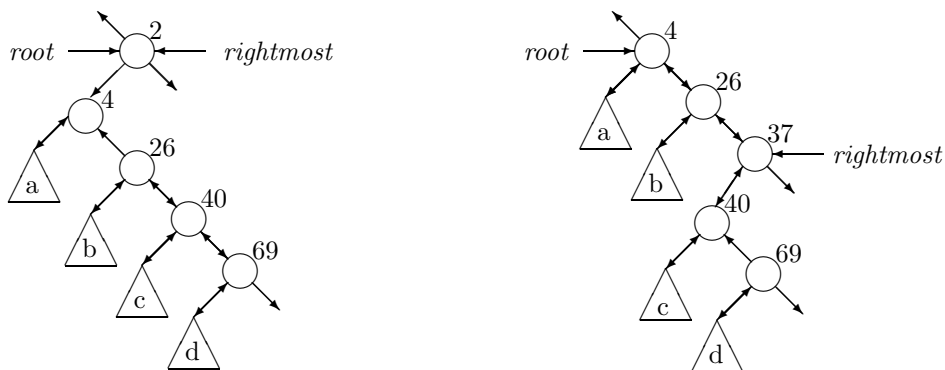
Spørgsmål 2A

Antag at en mængde er repræsenteret ved et binært træ i heap orden samt to pointere, som angivet ovenfor. Gør rede for hvordan $report(S, e)$ kan implementeres, så der højst indgår $2k + 1$ sammenligninger mellem elementer, hvis der rapporteres k elementer. □

$insert(S, e)$ implementeres ved at starte ved elementet, som $rightmost$ -pointeren peger på og følge up -pointerne mod roden indtil man møder en nil -pointer eller et element \hat{e} , der er mindre end e .

Hvis en nil -pointer mødes er e mindre end alle elementer i S så e gøres til ny rod, som får den gamle rod r som venstre efterfølger. $root$ og $rightmost$ sættes til at pege på e . Hvis man møder et element \hat{e} , der er mindre end e , gøres \hat{e} 's højre efterfølger til e 's venstre efterfølger hvorefter e gøres til en ny højre efterfølger af \hat{e} . $rightmost$ sættes igen til at pege på e . Se tegning nedenfor.

Følgende tegning viser resultatet af at henholdsvis $insert(T, 2)$ og $insert(T, 37)$ på træet T fra eksemplet på forrige side.



Spørgsmål 2B

Vis, at $init(S, e)$ efterfulgt af m $insert(S, \cdot)$ operationer kræver højst $2m$ sammenligninger mellem elementer. □

Opgave 3 (20%)

Lad $MOD_kDISTINCT(x_1, x_2, \dots, x_n)$ være afgørlighedsproblemet defineret ved:

Givet n reelle tal x_1, x_2, \dots, x_n .
Gælder der for alle $i \neq j$ at $(|j - i| \bmod k = 0) \Rightarrow (x_i \neq x_j)$?

Bemærk, at $MOD_1DISTINCT$ er identisk med problemet *element distinctness* fra noterne.

Spørgsmål 3A

Vis, at i den lineære beregningstræmodel er $\Omega(n \log n)$ en nedre grænse for problemet $MOD_2DISTINCT(x_1, x_2, \dots, x_n)$. □

Spørgsmål 3B

Vis, at i den lineære beregningstræmodel er $\Omega(n \log n)$ en nedre grænse for problemet $MOD_{\sqrt{n}}DISTINCT(x_1, x_2, \dots, x_n)$. □

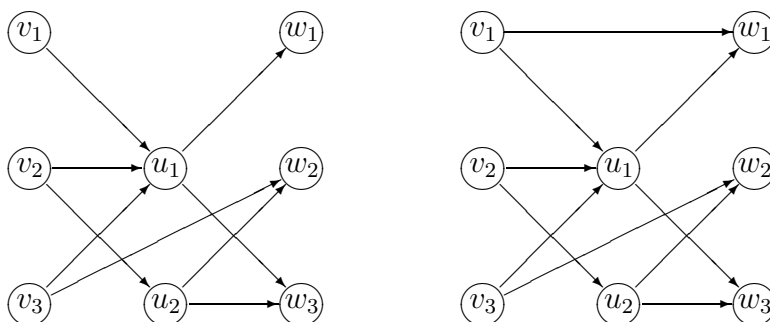
Opgave 4 (30%)

Lad $\vec{v} = (v_1, v_2, \dots, v_k)$ og $\vec{w} = (w_1, w_2, \dots, w_k)$ være to vektorer af samme størrelse k .

(G, \vec{v}, \vec{w}) en *router* fra \vec{v} til \vec{w} hvis og kun hvis følgende er opfyldt:

- $G = (V, E)$ er en orienteret graf.
- $\{v_1, v_2, \dots, v_k\}$ og $\{w_1, w_2, \dots, w_k\}$ er delmængder af V .
- Der findes der k knudedisjunkte veje der forbinder v_1 med w_1 , v_2 med w_2 ... v_k med w_k .

På følgende tegning er grafen til venstre *ikke* en router, mens den til højre er en router ((v_1, w_1) , $(v_2 \rightarrow u_2 \rightarrow w_2)$ og $(v_3 \rightarrow u_1 \rightarrow w_3)$ er tre knudedisjunkte veje fra v_1 til w_1 , v_2 til w_2 , v_3 til w_3).



ROUTER er problemet:

Givet en orienteret graf $G = (V, E)$ og vektorer $\vec{v} = (v_1, v_2, \dots, v_k)$ og $\vec{w} = (w_1, w_2, \dots, w_k)$ således at $\{v_1, v_2, \dots, v_k\}$ og $\{w_1, w_2, \dots, w_k\}$ er delmængder V . Er (G, \vec{v}, \vec{w}) en router?

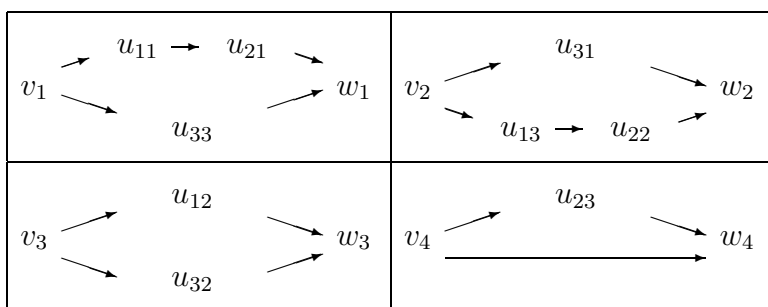
I det følgende defineres en afbildning F fra logiske udtryk i konjunktiv normalform med 3 literaler i hver klausul, til orienterede grafer.

På et logisk udtryk Φ med m klausuler over n variable kan $F(\Phi)$ beskrives som følger:

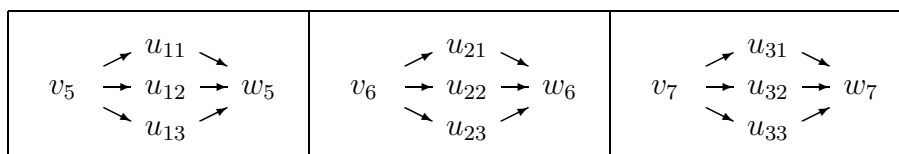
- For hver af de n variabler x_i ($i = 1, 2, \dots, n$) i Φ er der to knuder v_i og w_i i grafen $F(\Phi)$.
- For hver klausul C_j ($j = 1, 2, \dots, m$) i Φ er der ligeledes to knuder v_{n+j} og w_{n+j} i grafen $F(\Phi)$.
- Til hver af de $3m$ literaler, α_{js} ($j = 1, 2, \dots, m$ og $s = 1, 2, 3$) i Φ svarer en knude u_{js} i $F(\Phi)$.
- Mellem v_i og w_i svarende til variabel x_i er der to knudedisjunkte veje i $F(\Phi)$. Den ene passerer alle de knuder u_{js} , der svarer til forekomster af x_i i klausulerne. Den anden passerer alle de knuder u_{js} , der svarer til forekomster af \bar{x}_i i klausulerne.
- Mellem v_{n+j} og w_{n+j} svarende til klausul C_j er der tre knudedisjunkte veje af længde 2. De går gennem hver af de tre knuder u_{j1}, u_{j2} og u_{j3} .

Eksempel

Hvis $\Phi = (x_1 \vee x_3 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_1)$ er $F(\Phi)$ en graf med knuder $\{v_1, v_2, \dots, v_7\} \cup \{w_1, w_2, \dots, w_7\} \cup \{u_{11}, u_{12}, u_{13}, u_{21}, u_{22}, u_{23}, u_{31}, u_{32}, u_{33}\}$ og kanter, der giver anledning til følgende delgrafer af $F(\Phi)$:



og



Spørgsmål 4A

Vis, at F kan danne basis for en logspace-reduktion fra 3SAT til ROUTER og dermed at ROUTER er NP -fuldstændigt (NP -complete). Der kræves ingen argumentation for at F kan beregnes i logaritmisk plads. \square

En orienteret graf har dybde d , hvis den maksimale vejlængde i grafen er højst d . (Længden af en vej er antallet af kanter, der indgår i vejen).

d -ROUTER problemet er:

Givet en orienteret graf $G = (V, E)$ med dybde d samt vektorer $\vec{v} = (v_1, v_2, \dots, v_k)$ og $\vec{w} = (w_1, w_2, \dots, w_k)$ således at $\{v_1, v_2, \dots, v_k\}$ og $\{w_1, w_2, \dots, w_k\}$ er delmængder af V . Er (G, \vec{v}, \vec{w}) en router?

Spørgsmål 4B

Vis, at 3-ROUTER er NP -fuldstændigt. \square

Spørgsmål 4C

Vis, at 2-ROUTER er i P . \square