Eksamen i fag
SIF8010 Algoritmer og datastrukturer
Lørdag 9. august 2003, kl. 0900–1500

Faglig kontakt under eksamen:
Arne Halaas, tlf. 41661982; Magnus Lie Hetland, tlf. 91851949.
Hjelpemidler: Alle kalkulatortyper tillatt. Alle trykte og håndskrevne hjelpemidler tillatt.
Svar fortrinnsvis i anviste ruter på oppgavearket.
Tilleggsark kan vedlegges om nødvendig. Skriv studentnummer på alle ark.
Oppgaven består av i alt 6 ark.

Oppgave 1 (50%)

I de følgende tre deloppgavene (1 **a**, **b** og **c**) skal du bruke den vektede, rettede grafen $G = (V, E)$, med $V = \{1 \ldots 6\}$. Kant-vektene defineres av matrisen C, slik at $C[i, j]$ er vekten til kanten fra i til j.

Vektmatrisen C er implementert ved hjelp av nabolister som er lagret i den endimensjonale tabellen W, slik at $W[i]$ er nabolisten til node i. Nabolistene inneholder par av typen (w, j) der w er vekten på kanten til den aktuelle nabonoden j. Kanter som ikke eksplisitt er oppgitt antas å ha uendelig stor vekt.

I det følgende, anta at første element i alle tabeller har indeks 1. Tabellen W er definert slik:

$$
\begin{aligned}
W[1] &= [(4,2),(6,3),(2,4)] \\
W[2] &= [\,] \\
W[3] &= [(1,5),(1,6)] \\
W[4] &= [(2,5)] \\
W[5] &= [(4,6)] \\
W[6] &= [(3,2)]
\end{aligned}
$$

Ut fra denne definisjonen ser vi for eksempel at kanten fra node 1 til node 3 har vekt 6 ($C[1,3] = 6$) og at kanten fra node 2 til node 4 har vekt ∞ ($C[2,4] = \infty$).

Merknad til retting: I den opprinnelige eksamen sto det $C[1,4]$ i den siste parentesen over (noe som opplagt er galt).

Alle algoritmene i denne oppgaven skal følge prioritetsregelen gitt nedenfor:

> **Prioritetsregel**: Der en algoritme kan velge mellom flere noder, anta at den alltid velger den av de mulige nodene som har lavest nummer. Der en algoritme kan velge mellom flere kanter, anta at den alltid velger den av de mulige kantene som kommer tidligst i en leksikalsk sortering.

Dette betyr at node 3 velges før node 6, at kanten fra 2 til 3 velges før kanten fra 2 til 4, men etter kanten fra 1 til 5.

a (10%). Finn en topologisk sortering av grafen. Bruk dybde-først-søk til dette, som vist i læreboka, og følg **prioritetsregelen**. Bruk denne rekkefølgen til å finne korteste vei fra node 1 til node 2 med dynamisk programmering (DAG-SHORTEST-PATH). Bruk tabellen nedenfor til å vise hvordan hvert trinn i algoritmen oppdaterer $d[i]$ (avstandsestimatet til node i):

	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$
Start	0	∞	∞	∞	∞	∞
Trinn 1						
Trinn 2						
Trinn 3						
Trinn 4						
Trinn 5						

b (10%). Utfør Prims algoritme på G sin underliggende urettede graf det vil si, grafen som er lik G bortsett fra at kantene ikke har retning. (Kantenes opprinnelige retning kan fremdeles ha betydning for **prioritetsregelen**.) Oppgi kantene i den rekkefølgen de legges til i spenntreet på. Kantene beskrives med fra- og til-node som oppgitt i tabellen W. Start i node 1. Svar i tabellen nedenfor.

	Fra-node	Til-node
Trinn 1		
Trinn 2		
Trinn 3		
Trinn 4		
Trinn 5		

c (10%). Utfør Dijkstras algoritme på G for å finne korteste vei fra node 1 til alle andre noder. (Her har selvfølgelig kantenes retning stor betydning.) Vær nøye med å følge **prioritetsregelen**. Svar i tabellen nedenfor, på tilsvarende måte som i oppgave **a**. "Start" er tilstanden etter at initialiseringen har blitt utført, og "Trinn X" er tilstanden etter at X noder har blitt fargelagt grå og fått RELAX kjørt på sine naboer.

	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$
Start						
Trinn 1						
Trinn 2						
Trinn 3						
Trinn 4						
Trinn 5						
Trinn 6						

d (10%). Anta at du skal implementere Kruskals algoritme for grafer der kant-vektene er heltall i et fast tallområde $[0, k]$ for en liten, konstant verdi k. Du bestemmer deg for å bruke tellesortering på kantene først, slik at de blir tilgjengelige i riktig rekkefølge. Bortsett fra dette implementerer du algoritmen som normalt (som i læreboka). Hva blir kjøretiden? Uttrykk kjøretiden med Θ-notasjon, der m er antall kanter og n er antall noder.

Kjøretid:

e (10%). Anta rent hypotetisk at å flette (MERGE) to sorterte tabeller kunne gjøres i konstant tid. Hva ville da kjøretiden til flettesortering (MERGE-SORT) bli? Bruk Θ-notasjon.

Kjøretid:

Oppgave 2 (35%)

Anta at du har et grafikk-bibliotek tilgjengelig som lar deg tegne linjer i rutenett av typen vist i figur 1 og 2 på side 6. Kall til dette biblioteket gjøres med følgende pseudokode:

```
draw a line from (x1, y1) to (x2, y2);
```

Dette utsagnet tegner en rett linje fra punktet med koordinater `(x1, y1)` til punktet med koordinater `(x2, y2)`.

De følgende deloppgavene dreier seg om funksjonene `func1` og `func2`, beskrevet med pseudokode nedenfor (`abs` er absoluttverdi-funksjonen):

```
func1(double a, b, c, d, e, f) {
  if (abs(b-a) <= 1) return;
  double g = (a + b)/2;
  draw a line from (g, c) to (g, d);
  if (e > g) {
    func2(g, b, c, d, e, f);
  }
```

```
  if (e < g) {
    func2(a, g, c, d, e, f);
  }
}

func2(double a, b, c, d, e, f) {
  if (abs(d-c) <= 1) return;
  double g = (c + d)/2;
  draw a line from (a, g) to (b, g);
  if (f > g) {
    func1(a, b, g, d, e, f);
  }
  if (f < g) {
    func1(a, b, c, g, e, f);
  }
}
```

a (5%). Selv om funksjonene `func1` og `func2` i utgangspunktet ikke gjør noe nyttig, minner de (tilsammen) om en todimensjonal variant av en algoritme i pensum. Hvilken?

Algoritme:

b (10%). Tegn resultatet av å kjøre `func1(0, 160, 0, 160, 160, 150)` i rutenettet i figur 2.

c (10%). Tegn resultatet av å kjøre `func1(0, 160, 0, 160, 70, 50)` i rutenettet i figur 1.

d (10%). Anta at vi setter $m =$ `b-a` og $n =$ `d-c`. Uttrykk kjøretiden til `func1` som funksjon av m og n. Velg selv hva du mener er mest fornuftig av *best-case* eller *worst-case* kjøretid (sett ett kryss). Bruk Θ-notasjon.

Kjøretid: ☐ Best-case ☐ Worst-case

Oppgave 3 (15%)

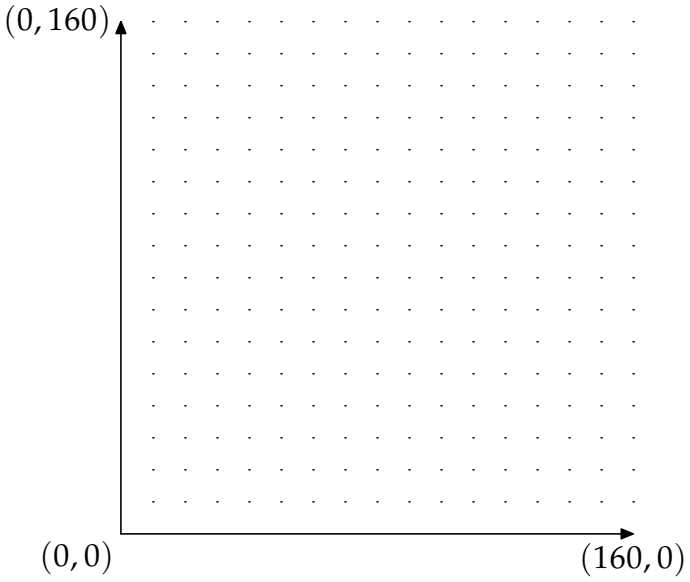
N forskjellige heltall settes inn i et binært søketre i tilfeldig rekkefølge. La $Q(N)$ betegne den gjennomsnittlige (forventede) dybden til det minste elementet i treet. Spesielt er $Q(0) = 0$, $Q(1) = 1$ og $Q(2) = 1.5$.

a (5%). Utled en rekurrensformel for $Q(N)$, der $N > 0$. Svar i ruten nedenfor.

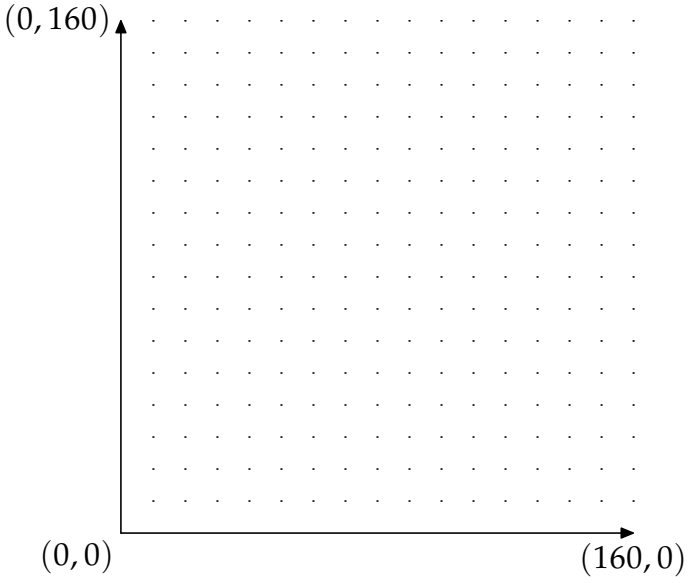
$Q(N) =$

b (10%). Bruk rekurrensformelen fra **a** til å finne et eksplisitt uttrykk for $Q(N)$ (dvs. uten bruk av Q til høyre for likhetstegnet). Svar i ruten nedenfor.

$Q(N) =$



Figur 1: Rutenett



Figur 2: Rutenett