Skriftlig Eksamen
Algoritmer og Datastrukturer (DM02)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Tirsdag den 24. januar 2006, kl. 9–13

Alle sædvanlige hjælpemidler (lærebøger, notater, osv.) samt brug af lommeregner er tilladt.

Eksamenssættet består af 4 opgaver på 6 nummererede sider (1–6).
Fuld besvarelse er besvarelse af alle 4 opgaver.
De enkelte opgavers vægt ved bedømmelsen er angivet i procent.

Der må gerne refereres til algoritmer og resultater fra lærebogen inklusive øvelsesopgaverne. Henvisninger til andre bøger accepteres ikke som besvarelse af et spørgsmål.

Bemærk, at hvis der er et spørgsmål, man ikke kan besvare, må man gerne besvare de efterfølgende spørgsmål og blot antage, at man har en løsning til de foregående spørgsmål.

Husk at begrunde dine svar!

Opgave 1 (20%)

Spørgsmål a (7%): Dette spørgsmål handler om hashing med åben adressering. Betragt følgende hashtabel med 11 pladser

0	1	2	3	4	5	6	7	8	9	10
			14				3		41	

og følgende hash-funktion.

$$h(k,i) = \big(h_1(k) + ih_2(k)\big) \bmod 11,$$

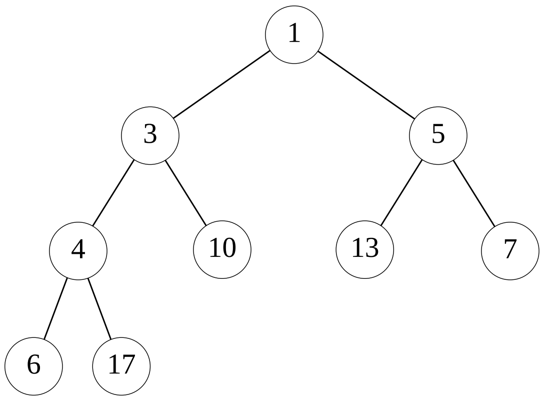
hvor

$$h_1(k) = k \bmod 11$$
$$h_2(k) = 1 + (k \bmod 10)$$

Som det ses ovenfor, er der allerede indsat tre elementer i hashtabellen.

Nu indsættes et element med nøgle 18. På hvilken plads havner dette element? □

Spørgsmål b (6%): Betragt følgende binære hob (binary heap).



Nu indsættes et element med prioritet 2. Tegn hoben, som den ser ud efter denne indsættelse. □

Spørgsmål c (7%): Angiv den asymptotiske løsning til rekursionsligningen

$$T(n) = 3T\left(\frac{n}{3}\right) + n^2 .$$

□

Opgave 2 (25%)

Denne opgave handler bl.a. om at tælle inversioner i en følge S af tal. En inversion er et par af tal x, y, hvor $x > y$, og x kommer før y i S.

Eksempel: I følgen $\langle 2, 5, 10, 3, 17, 12 \rangle$ er der tre inversioner: (5,3), (10,3) og (17,12).

Spørgsmål a (6%): Hvad er det største antal inversioner, der kan være i en følge af n tal?
Hvordan ser en følge med dette antal inversioner ud? □

Spørgsmål b (7%): Forklar, hvordan man kan modificere mergesort-algoritmen, så den beregner antallet af inversioner i en følge af tal. Køretiden skal forblive $O(n \log n)$.

Du må gerne antage, at alle tal i følgen er forskellige.

Bemærk, at algoritmen ikke skal opremse inversionerne, kun angive *antallet* af inversioner. □

Spørgsmål c (12%): Hvilken algoritme ville du anvende til at sortere en følge af n tal i de følgende tre tilfælde?

1. Følgen har $O(n)$ inversioner.
2. Følgen har $O(n \log n)$ inversioner, og tallene ligger mellem 0 og $n^2 - 1$.
3. Følgen har $O(n^2)$ inversioner.

Angiv i hvert tilfælde en algoritme, hvis asymptotiske worst-case køretid er så lille som muligt. □

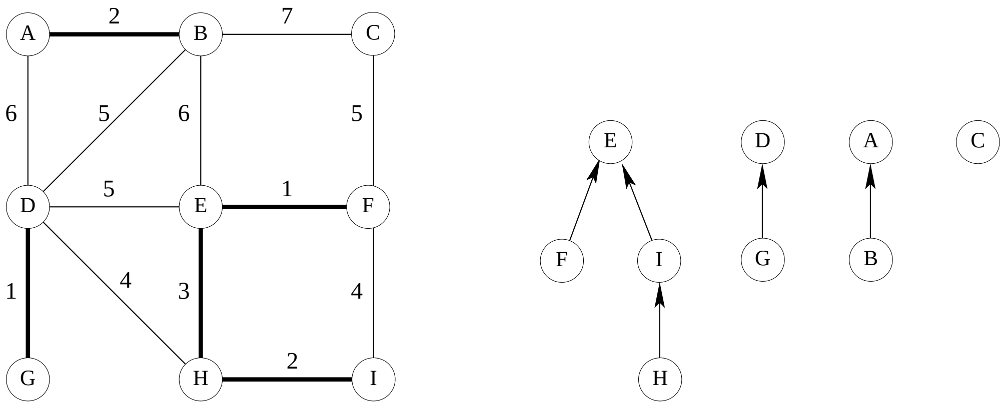
Opgave 3 (30%)

Denne opgave handler om vægtede grafer. De første tre spørgsmål handler om letteste udspændende træer, og det sidste handler om korteste veje.

Spørgsmål a (8%): I dette spørgsmål ser vi på Kruskals algoritme, realiseret vha. disjunkte mængder implementeret som træer. Der anvendes "union by rank" og "path compression".

I Figur 1(a) nedenfor er vist en vægtet graf. Kanterne tegnet med fed er de første 5 kanter, som Kruskals algoritme inkluderer i et letteste udspændende træ. Figur 1(b) viser skoven, der repræsenterer de fire disjunkte mængder, efter at disse 5 kanter er blevet valgt.

Hvilken kant er den næste, som Kruskals algoritme inkluderer i det udspændende træ?
Vis skoven, som repræsenterer de disjunkte mængder, efter at denne sjette kant er tilføjet. Husk at bruge union by rank og path compression. ☐



(a) En vægtet graf. De første 5 kanter, som Kruskals algoritme vil vælge, er tegnet med **fede** linier.

(b) De disjunkte mængder svarende til kanterne valgt i grafen til venstre.

Figur 1: Eksemplet brugt i Opgave 3 a,b,d

Spørgsmål b (6%): Nu ser vi på Prims algoritme.

Angiv samtlige kanter i et letteste udspændende træ for grafen i Figur 1(a). Kanterne skal angives i den rækkefølge, som Prims algoritme vil vælge dem i, hvis den starter i knuden A. □

Spørgsmål c (8%): Lad G være en vægtet graf, og lad $\{e_1, \ldots, e_k\}$ være kanter i G.

Forklar, hvordan man kan undersøge, om der findes et letteste udspændende træ for G, som indeholder disse k kanter.
Hvad er køretiden af din algoritme? □

Spørgsmål d (8%): Nu ser vi på Dijkstras algoritme.

Vis korteste-vej-træet, som fremkommer, når man kører Dijkstras algoritme på grafen i Figur 1(a) med udgangspunkt i knuden A.
Angiv også længden af en korteste vej fra A til hver af knuderne i grafen. □

Opgave 4 (25%)

Denne opgave handler om dynamisk programmering.

Lad x, y og z være strenge af længde n, m og $n+m$. Strengen z kaldes et *flet* af x og y, hvis man kan opnå z ved at flette x og y.

Eksempel: **gul**e**r**o**d** er et flet af **gled** og uro
datastrukturer er et flet af **data** og strukturer
abe**k**a**t**te er et flet af **akt** og beate

Følgende rekursive algoritme afgør, om z er et flet af x og y.

```
StrengFlet(x, y, z)
    hvis |x| + |y| ≠ |z|
        returner Falsk
    returner Flet(x, y, z, |x|, |y|)

Flet(x, y, z, n, m)
    hvis (m = 0 ∧ n = 0)
        returner Sand
    hvis m = 0
        returner Fx(n, 0)
    hvis n = 0
        returner Fy(0, m)
    returner (Fx(n, m)  ∨  Fy(n, m))

Fx(x, y, z, i, j)
    returner (z[i+j] = x[i]  ∧  Flet(x, y, z, i−1, j))

Fy(x, y, z, i, j)
    returner (z[i+j] = y[j]  ∧  Flet(x, y, z, i, j−1))
```

Husk, at $\wedge$ betyder "og", og $\vee$ betyder "eller". For enhver streng s betegner $s[i]$ det i'te tegn i s, og $|s|$ betyder længden af s.

Spørgsmål a (8%): Indsæt sandhedsværdierne for FLET(“akt”, “beate”, “abekatte”, i, j) i følgende tabel. Du kan evt. bruge kopien af tabellen, som optræder på næste side.

		j					
		0	1	2	3	4	5
i	0						
	1						
	2						
	3						

□

Spørgsmål b (8%): Hvor mange kald til FLET(“akt”, “beate”, “abekatte”, 2, 3) foretages der, hvis man udfører FLETSTRENG(“akt”, “beate”, “abekatte”)? □

Spørgsmål c (9%): Skriv i pseudokode en *iterativ* algoritme, som vha. *dynamisk programmering* afgør, om en streng z er et flet af to strenge x og y.
Hvad er køretiden af din algoritme? □

		j					
		0	1	2	3	4	5
i	0						
	1						
	2						
	3						