Skriftlig Eksamen
Algoritmer og Datastrukturer (DM507)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Mandag den 7. juni 2010, kl. 9–13

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

Eksamenssættet består af 5 opgaver på 8 nummererede sider (1–8).
Fuld besvarelse er besvarelse af alle 5 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 (10%)

Spørgsmål a (5%): Løs følgende rekursionsligning.

$$T(n) = 16 \cdot T(n/2) + n^4 + n^2$$

□

Spørgsmål b (5%):

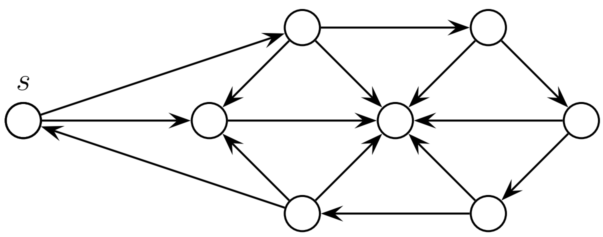
Angiv et Huffman-træ for en streng med følgende alfabet og tilhørende hyppigheder.

Tegn	`a`	`b`	`c`	`d`	`e`	`f`	`g`
Hyppighed	300	150	75	125	200	50	100

□

Opgave 2 (25%)

Spørgsmål a (6%): For alle knuder v i grafen G_1, angiv distanceværdien $v.d$ som tildeles ved bredde-først søgning (BFS) med start i knuden s.



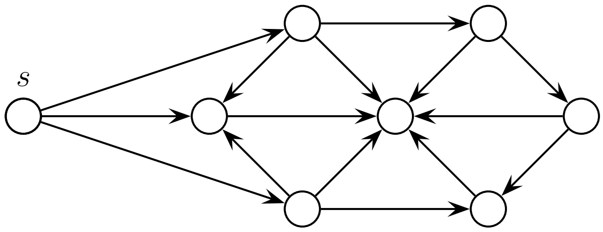
Figur 1: Grafen G_1

Sidst i sættet er der en kopi af graferne i denne opgave. Dem kan du evt. bruge under besvarelsen (husk i så fald at aflevere siden sammen med resten af din besvarelse).

□

Spørgsmål b (7%): For alle knuder v i grafen G_2, angiv starttiden (discovery time) $v.d$ og sluttiden (finishing time) $v.f$ som tildeles ved dybde-først søgning (DFS) med start i knuden s.

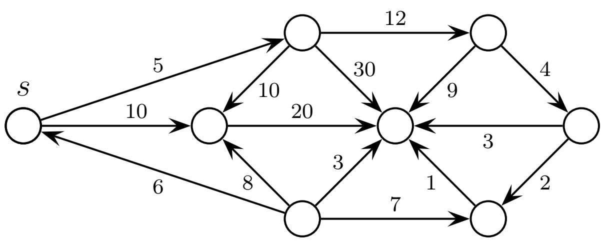
(For DFS afhænger det præcise resultat af ordningen af knuders nabolister. Du skal her antage at på figuren er alle knudes nabolister ordnet “med uret”, startende fra “lodret opad”.)



Figur 2: Grafen G_2

□

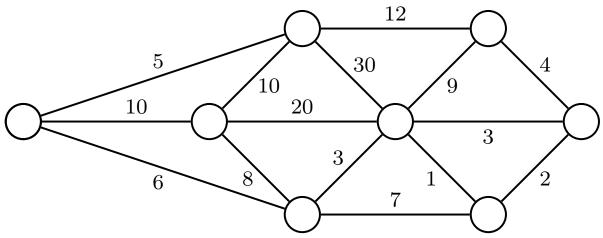
Spørgsmål c (6%): For alle knuder v i grafen G_3, angiv distanceværdien $v.d$ som tildeles ved kørsel af Dijkstras algoritme med start i knuden s.



Figur 3: Grafen G_3

□

Spørgsmål d (6%): For grafen G_4, angiv et minimum spanning tree (MST), samt dets vægt.



Figur 4: Grafen G_4

□

Opgave 3 (25%)

Som bekendt skrives tallet 43 i 2-talssystemet som 101011 fordi

$$43 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

Vi ønsker for et vilkårligt positivt heltal n at finde cifrene $b_k b_{k-1} \dots b_1 b_0$ i n's repræsentation i 2-talssystemet. For eksempel, for $n = 43$ er $b_5 b_4 \dots b_1 b_0$ lig 101011.

Betragt følgende algoritme til at generere k og b_i'er:

BinaryDigits(n)
  $i = 0$
  $d = n$
  while $d > 0$
    $b_i = d \bmod 2$
    $d = d \text{ div } 2$
    $i = i + 1$
  $k = i - 1$

Her er "div" heltalsdivision, og "mod" er rest ved heltalsdivision. For eksempel har vi 9 div 2 = 4 og 9 mod 2 = 1. Der gælder altid

$$x = y \cdot (x \text{ div } y) + (x \text{ mod } y) \tag{1}$$

for alle hele tal x og y.

Spørgsmål a (6%):

Angiv resultatet (dvs. k og b_i'erne) af ovenstående algoritme når input er $n = 55$.

☐

Spørgsmål b (9%):

Vis følgende invariant for **while** -løkken:

Når testen ved indgangen til **while** -løkken udføres, gælder

$$\begin{array}{rl} \text{i)} & n = d \cdot 2^i + \sum_{j=0}^{i-1} b_j \cdot 2^j \\ \text{ii)} & d \geq 0 \end{array}$$

(For $i = 0$ er summen tom, og har pr. definition værdien nul.)

Hint: brug (1) for passende x og y.

☐

Spørgsmål c (6%):

Vis at algoritmen er korrekt, dvs. at den for alle positive heltal n beregner n's repræsentation $b_k b_{k-1} \cdots b_1 b_0$ i 2-talssystemet.

☐

Spørgsmål d (4%):

Giv en analyse af køretiden for algoritmen som funktion af n.

☐

Opgave 4 (25%)

Denne opgave handler om at finde en største fælles *vægtet* delsekvens af to strenge. Her har hvert tegn c i alfabetet en vægt $w(c)$ tilknyttet. Et eksempel kunne være alfabetet

Tegn c	`a`	`b`	`c`	`d`
Vægt $w(c)$	2	4	1	3

Vægten af en fælles delsekvens er summen af vægtene af tegnene i delsekvensen. F.eks. er `acd` en delsekvens af strengene $X =$ `acbd` og $Y =$ `bacda`, og dens vægt er $2 + 1 + 3 = 6$, hvis alfabetets vægte er som angivet ovenfor.

Givet to strenge $X = x_1 x_2 \cdots x_n$ og $Y = y_1 y_2 \cdots y_m$ (af længde henholdsvis n og m) over et vægtet alfabet ønsker vi at finde den største vægt en fælles delsekvens af X og Y kan have.

Mere generelt lader vi, for $0 \leq i \leq n$ og $0 \leq j \leq m$, $W(i,j)$ betegne den største vægt en fælles delsekvens af strengene $X = x_1 x_2 \cdots x_i$ og $Y = y_1 y_2 \cdots y_j$ kan have. Svaret på det oprindelige problem er så $W(n,m)$.

$W(i,j)$ kan beskrives ved følgende rekursive formel:

$$W(i,j) = \begin{cases} 0 & \text{hvis } i = 0 \vee j = 0 \\ \max\{W(i-1,j), W(i,j-1)\} & \text{hvis } i,j \geq 1 \wedge x_i \neq y_j \\ \max\{W(i-1,j), W(i,j-1), & \\ \qquad W(i-1,j-1) + w(x_i)\} & \text{hvis } i,j \geq 1 \wedge x_i = y_j \end{cases}$$

Spørgsmål a (9%):

Udfyld for strengene $X =$ `acbd` og $Y =$ `bacda` og ovenstående vægte for alfabetet tabellen for $W(i,j)$ i Figur 5.

Sidst i sættet er der en kopi af tabellen i dette spørgsmål. Den kan du evt. bruge under besvarelsen (husk i så fald at aflevere siden sammen med resten af din besvarelse).

□

$i \setminus j$	0	1	2	3	4	5
0						
1						
2						
3						
4						

Figur 5: Tabel over $W(i, j)$

Spørgsmål b (8%):

Beskriv en algoritme baseret på dynamisk programmering der givet X, Y og et vægtet alfabet beregner den største vægt en delsekvens af X og Y kan opnå. Analyser algoritmens køretid.

□

Spørgsmål c (8%):

Argumentér for at den rekursive formel for $W(i, j)$ er korrekt.

□

Opgave 5 (15%)

I denne opgave ønsker vi at udvide binære søgetræer med oplysninger om afstandene mellem de gemte nøgler, og specielt ønsker vi at kunne finde den største afstand i træet mellem en nøgle og dens predecessor.

Mere præcist, hvis et søgetræ gemmer n nøgler $x_1 \leq x_2 \leq x_3 \leq \cdots \leq x_n$, er de søgte afstande $(x_2 - x_1), (x_3 - x_2), \cdots, (x_n - x_{n-1})$, og vi ønsker at kunne finde den største af disse. Hvis eksempelvis nøglerne gemt i træet er 3, 5, 11, 14, 23 og 30, er afstandene 2, 6, 3, 9 og 7, og største afstand er 9, som opnås mellem nøglen 23 og den predecessor 14.

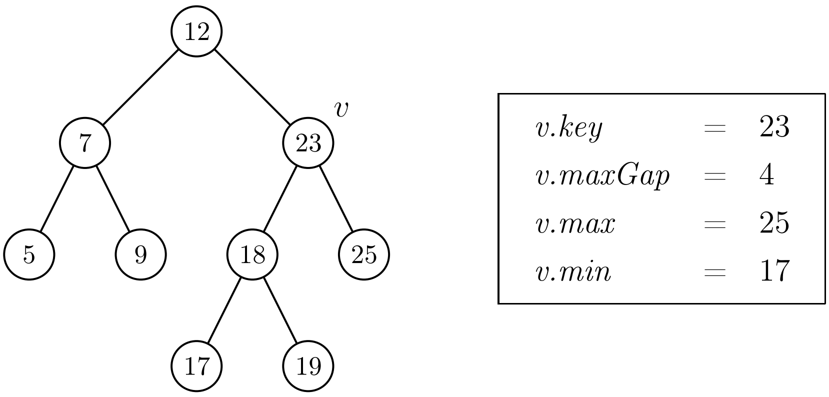
Vi udstyrer nu enhver knude v i søgetræet med følgende tre ekstra informationer (udover den i knuden gemte nøgle $v.key$):

1. Største afstand mellem nøgler gemt i v's undertræ ($v.maxGap$).
2. Største nøgle gemt i v's undertræ ($v.max$).
3. Mindste nøgle gemt i v's undertræ ($v.min$).

(Husk at en knudes undertræ inkluderer knuden selv. Hvis der kun er én nøgle i v's undertræ, sættes $v.maxGap$ lig nul.)

Specielt kan største afstand i træet herved aflæses af roden r's information $r.maxGap$ i $O(1)$ tid.

Et eksempel på et binært søgetræ og informationen i en af dets knuder kan ses i Figur 6.



Figur 6: Eksempel på informationen i en knude

Spørgsmål a (6%):

Angiv hvordan en knudes informationer kan bestemmes i $O(1)$ tid ud fra informationerne i knudens to børn, samt knudens og børnenes nøgler.

(Et eller begge af børnene kan være NIL, hvilket giver (simple) specialtilfælde som du ikke behøver beskrive).

□

Vi lader nu søgetræet være et rød-sort træ.

Spørgsmål b (4%):

Argumentér for at informationerne i træets knuder kan vedligeholdes under indsættelser og sletninger, uden at ændre køretiden $O(\log n)$ for disse.

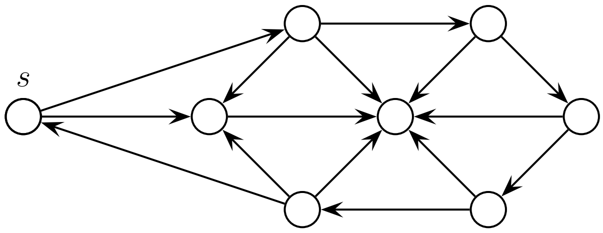
□

Som sagt kan man i $O(1)$ tid finde den største afstand i træet mellem nøgler og deres predecessorer ved at aflæse roden r's information $r.maxGap$. Vi ønsker nu også at kunne finde en konkret nøgle i træet som har denne afstand til sin predecessor.

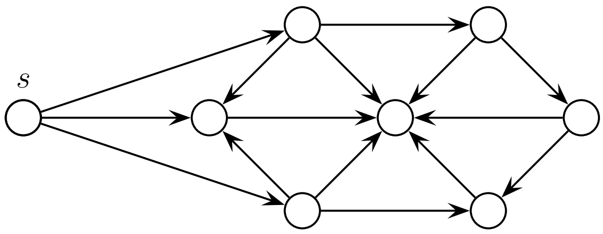
Spørgsmål c (5%):

Beskriv en søgeproces som i $O(\log n)$ tid finder en sådan nøgle.

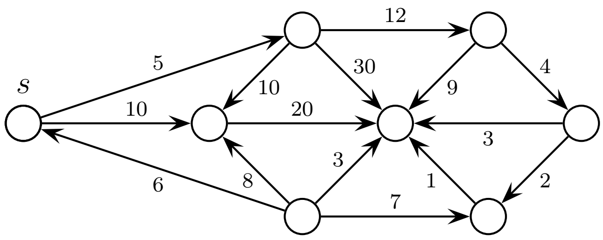
□



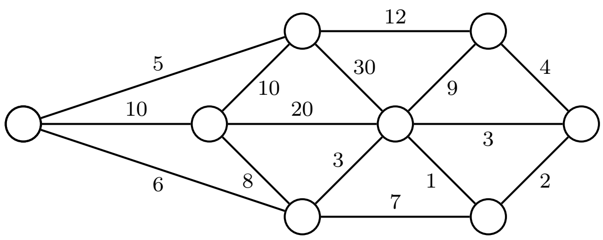
Figur 7: Grafen G_1 (til afleveringsbrug)



Figur 8: Grafen G_2 (til afleveringsbrug)



Figur 9: Grafen G_3 (til afleveringsbrug)



Figur 10: Grafen G_4 (til afleveringsbrug)

$i \setminus j$	0	1	2	3	4	5
0						
1						
2						
3						
4						

Figur 11: Tabel over $W(i, j)$ (til afleveringsbrug)