Skriftlig Eksamen
DM507 Algoritmer og Datastrukturer

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Mandag den 8. juni 2015, kl. 16:00–20:00

Besvarelsen skal afleveres elektronisk. Se vejledning udsendt i kurset.

Alle skriftlige hjælpemidler (lærebøger, notater, osv.) samt brug af computer er tilladt. Det er ikke tilladt at bruge internettet, undtagen til den elektroniske aflevering.

Eksamenssættet består af 10 opgaver på 10 nummererede sider (1–10). Fuld besvarelse er besvarelse af alle 10 opgaver. De enkelte opgavers vægt ved bedømmelsen er angivet i procent.

Der må gerne refereres til algoritmer og resultater fra lærebogen (Cormen et al., *Introduction to Algorithms*, tredie udgave), samt andre materialer fra kurset (f.eks. opgavesedler og slides). Henvisninger til andre kilder kan ikke bruges i besvarelsen 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.

Opgave 1 (10%)

Denne opgaver handler om følgende rekursionsligninger:

$$
\begin{array}{rl}
\text{i)} & T(n) = 5 \cdot T(n/2) + n^2 \\
\text{ii)} & T(n) = 5 \cdot T(n/5) + n \log n \\
\text{iii)} & T(n) = 2 \cdot T(n/5) + n^{1/2} \\
\text{iv)} & T(n) = 2 \cdot T(n-2) + n
\end{array}
$$

Spørgsmål a (5%):

Angiv for hver rekursionsligning *om* den kan løses med Master Theorem fra lærebogen.

Spørgsmål b (5%):

Angiv løsningen $T(n)$ for hver rekursionsligning som kan løses med Master Theorem.

Opgave 2 (10%)

Angiv for hvert af nedenstående udsagn, om de er sande eller falske.

$$
\begin{array}{rl}
\text{i)} & 2^n \text{ er } O(n^3) \\
\text{ii)} & n^2 \text{ er } O(3^n) \\
\text{iii)} & n(\log n)^2 \text{ er } O(n^3 \log n) \\
\text{iv)} & n^2 \log n \text{ er } O(n(\log n)^3) \\
\text{v)} & n^3 \text{ er } O(n^2) \\
\text{vi)} & 3^n \text{ er } O(2^n) \\
\text{vii)} & n^{1/3} \text{ er } O(n^{1/2}) \\
\text{viii)} & (1/3)^n \text{ er } O((1/2)^n) \\
\text{ix)} & 3 \text{ er } O(2)
\end{array}
$$

Opgave 3 (7%)

Udfør først HEAP-INCREASE-KEY(A,9,15) og dernæst HEAP-EXTRACT-MAX(A) på nedenstående max-heap A.

Angiv efter hver af de to operationer heapens udseende ved at skrive elementerne i rækkefølge fra venstre mod højre.

	1	2	3	4	5	6	7	8	9
A:	18	9	16	4	8	12	13	1	2

Opgave 4 (7%)

Nedenstående er en hashtabel H der bruger quadratic probing, med auxiliary hashfunktion $h'(x) = (3x + 5) \bmod 11$ og med konstanter $c_1 = 3$ og $c_2 = 1$.

	0	1	2	3	4	5	6	7	8	9	10
H:	13	39		36					23	5	

Indsæt værdierne 22, 16 og 17 (i den rækkefølge). Angiv udseendet af hashtabellen efter hver af de tre indsættelser.

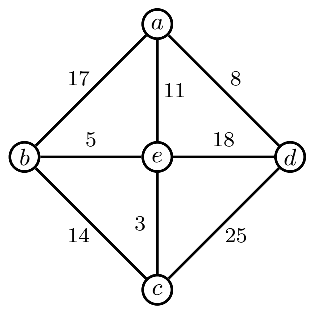
Svar for hver indsættelse ved at skrive indholdet af H i rækkefølge fra venstre mod højre, med tomme pladser angivet som `x`.

Opgave 5 (7%)

Udfør Prims algoritme på grafen nedenfor, med start i knuden a.

Som svar, angiv knuderne i den rækkefølge de udtages fra prioritetskøen (med operationen EXTRACT-MIN) under algoritmens kørsel.

Angiv også kanterne i det resulterende minimum spanning tree (MST). En kant med endepunkter u og v skrives som sædvanligt (u, v).

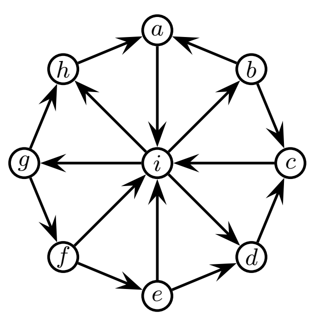


Opgave 6 (10%)

Spørgsmål a (6%):

For alle knuder v i grafen nedenfor, angiv starttiden (discovery time) $v.d$ og sluttiden (finishing time) $v.f$ som tildeles ved dybde-først søgning (DFS) med start i knuden i.

For DFS afhænger resultatet af ordningen af knuders nabolister. Du skal her antage at en knudes naboliste er sorteret i alfabetisk orden efter naboknudernes navne.



Spørgsmål b (4%):

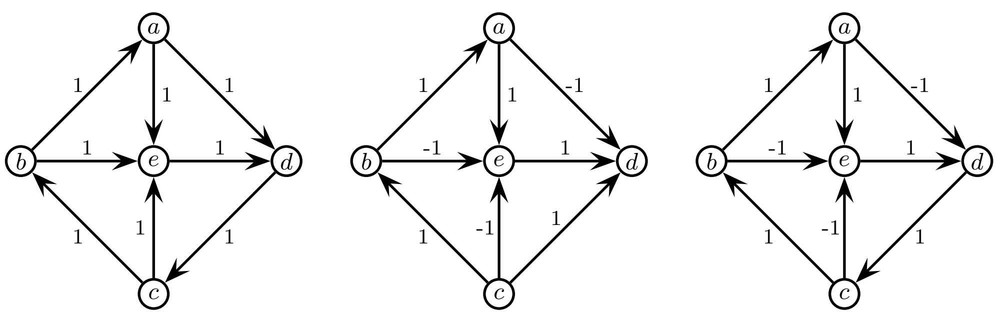
For alle kanter i grafen ovenfor, bestem deres type (tree edge, back edge, forward edge, cross edge) ved DFS gennemløbet fra spørgsmål **a** (dvs. DFS gennemløb med start i knude i).

Svar ved at angive fire lister af kanter, én liste for hver kanttype. En kant med endepunkter u og v skrives som sædvanligt (u, v). I hver kant, angiv endepunkterne i alfabetisk rækkefølge.

Opgave 7 (10%)

Spørgsmål a (6%):

For hvilke af nedenstående grafer kan de korteste veje findes med algoritmerne Dijkstra (D), Bellman-Ford (BF), DAG-Shortest-Paths (DAG), Breath-First-Search (BFS) og Depth-First-Search (DFS)?



G_1 G_2 G_3

Angiv svaret med en tabelopstilling som nedenstående. Sæt kryds i en tabelindgang når svaret er ja.

	D	BF	DAG	BFS	DFS
G_1					
G_2					
G_3					

Spørgsmål b (4%):

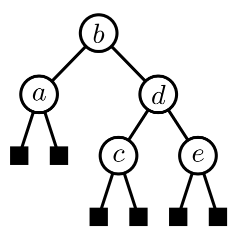
Vi ser nu på grafer med n knuder og $m = n \log n$ kanter.

Angiv for hver af algoritmerne DIJKSTRA, BELLMAN-FORD og DAG-SHORTEST-PATHS deres asymptotiske køretid som funktion af n på sådanne grafer (det antages at algoritmerne kan bruges på de pågældende grafer).

Opgave 8 (10%)

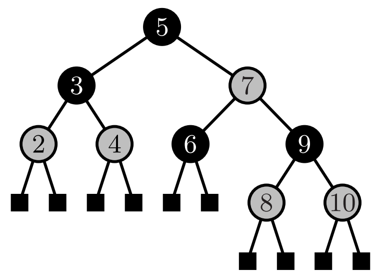
Spørgsmål a (5%):

Angiv en farvning af knuderne i træet nedenfor som gør det til et rød-sort træ.

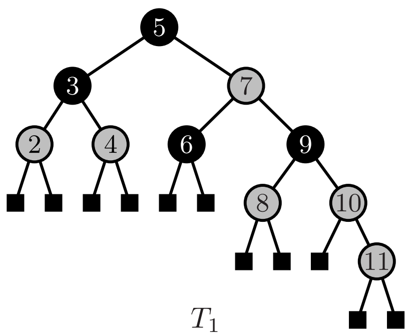


Svar ved at skrive en liste af navnene på de sorte knuder og en liste med navnene på de røde knuder.

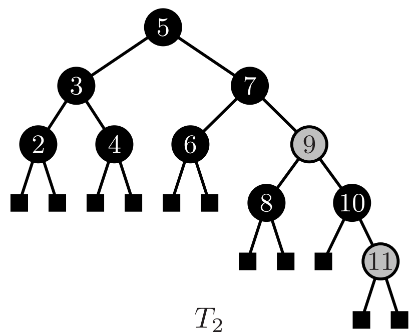
Spørgsmål b (5%):



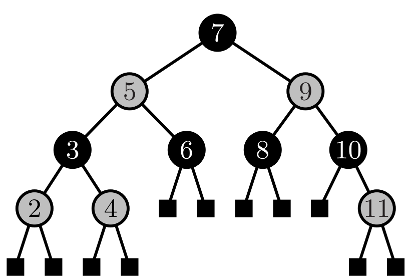
I ovenstående rød-sorte træ indsættes 11 under brug af algoritmen fra lærebogen. Angiv hvilket af nedenstående fire træer som er resultatet.



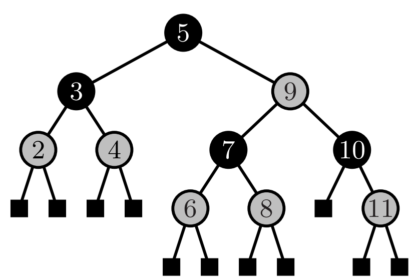
T_1



T_2



T_3



T_4

Opgave 9 (14%)

Angiv for hver af følgende algoritmer deres asymptotiske køretid i Θ-notation som funktion af n.

```
ALGORITME1(n)
    i = n
    while i > 1
        j = n
        while j > i
            j = j − 1
        i = i − 1
```

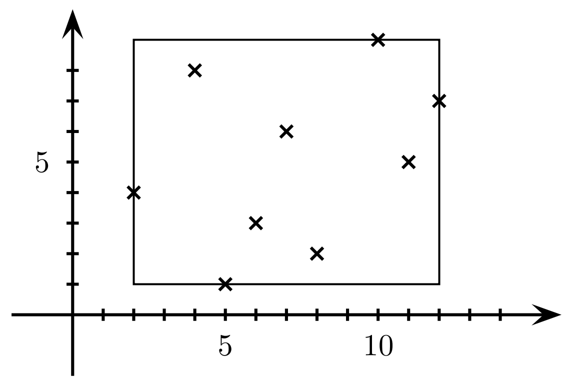
```
ALGORITME2(n)
    i = 1
    j = 1
    while i ≤ n
        while j ≤ i
            j = j + 1
        i = i ∗ 2
```

```
ALGORITME3(n)
    i = 1
    while i ≤ n
        j = 1
        while j ≤ i
            j = j + 1
        i = i ∗ 2
```

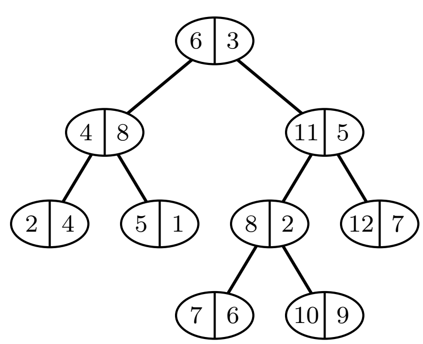
```
ALGORITME4(n)
    i = 1
    while i ≤ n
        j = 1
        while j ≤ i
            j = j ∗ 2
        i = i + 1
```

Opgave 10 (15%)

I denne opgave ønsker vi at gemme en mængde punkter i planen, og kunne svare på spørgsmål om det akse-parallelle rektangel, de udspænder. Et eksempel på en mængde punkter og deres udspændte rektangel er illustreret herunder.



Vi gemmer punkterne i knuderne i et søgetræ, hvor vi bruger punkternes x-koordinater som søgenøgle. Det må antages at ingen x-koordinat optræder i to forskellige punkter. Et eksempel på et sådant træ er vist herunder. Punkternes x-koordinater (træets søgenøgler) er angivet til venstre i knuderne, punkternes y-koordinater er angivet til højre.



Vi udstyrer nu enhver knude v i søgetræet med følgende fire ekstra informationer (udover de i knuden gemte x- og y-koordinater):

$$
\begin{aligned}
v.xmax &= \text{største } x\text{-værdi i } v\text{'s undertræ} \\
v.xmin &= \text{mindste } x\text{-værdi i } v\text{'s undertræ} \\
v.ymax &= \text{største } y\text{-værdi i } v\text{'s undertræ} \\
v.ymin &= \text{mindste } y\text{-værdi i } v\text{'s undertræ}
\end{aligned}
$$

(Husk at en knudes undertræ inkluderer knuden selv.)

Som eksempel vil det højre barn af roden i træet ovenfor have følgende informationer:

$$
\begin{aligned}
v.xmax &= 12 \\
v.xmin &= 7 \\
v.ymax &= 9 \\
v.ymin &= 2
\end{aligned}
$$

Spørgsmål a (3%):

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

(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 (3%):

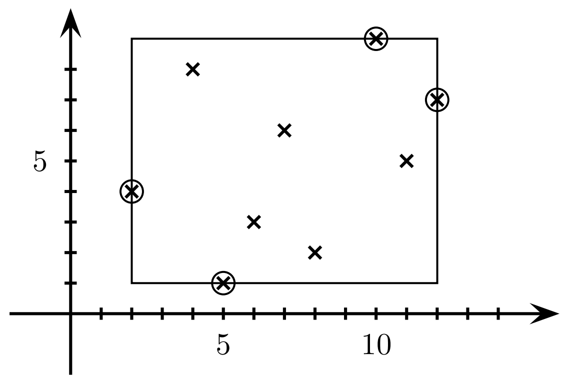
Argumentér for at informationerne i træets knuder kan vedligeholdes under indsættelser og sletninger, uden at køretiden $O(\log n)$ for indsættelser og sletninger ændres.

Spørgsmål c (3%):

Angiv hvordan arealet af det rektangel, som de gemte punkter punkterne udspænder, kan bestemmes i $O(1)$ tid ud fra træet.

Spørgsmål d (3%):

Vi ønsker nu for hver af de fire sider af det udspændte rektangel at finde et punkt som ligger på denne side. Som illustration ønsker vi i eksemplet fra starten af opgaven at finde de fire punkter fremhævet nedenfor.



Angiv hvordan man i $O(\log n)$ tid kan gøre dette ved hjælp af træet.

Spørgsmål e (3%):

Vi vil nu gerne opnå resultaterne i spørgsmål **c** (dvs. kunne finde arealet af rektangel i $O(1)$ tid) og **d** (dvs. kunne finde fire punkter, et på hver side af rektanglet, i $O(\log n)$ tid) på en anden måde. Indsættelser og sletninger skal stadig kunne foretages i $O(\log n)$ tid.

Angiv hvordan dette kan opnås ved at bruge to træer (uden ekstra information i alle knuder) i stedet for ovenstående datastruktur. Det må antages at heller ingen y-koordinat optræder i to forskellige punkter.