INSTITUT FOR DATALOGI, AARHUS UNIVERSITET

Science and Technology
EKSAMEN
Grundkurser i Datalogi
Algoritmer og Datastrukturer 1 (2003-ordning)
Antal sider i opgavesættet (incl. forsiden): 12
Eksamensdag: Fredag den 5. juni 2015, kl. 9.00-11.00
Tilladte medbragte hjælpemidler: Alle sædvanlige hjælpemidler (lærebøger og notater). Computer må ikke medbringes.
Materiale der udleveres til eksaminanden:

Årskort ____________________

Navn ____

Skriftlig Eksamen
Algoritmer og Datastrukturer 1 (2003-ordning)

Institut for Datalogi
Aarhus Universitet

Fredag den 5. juni 2015, kl. 9.00-11.00

Dette eksamenssæt består af en mængde multiple-choice-opgaver. Opgaverne besvares på opgaveformuleringen **som afleveres**.

For hver opgave er angivet opgavens andel af det samlede eksamenssæt.

Hvert delspørgsmål har præcist et rigtigt svar. For hvert delspørgsmål, kan du vælge **max ét svar** ved at afkrydse den tilsvarende rubrik. Et delspørgsmål bedømmes som følgende:

- Hvis du sætter kryds ved det rigtige svar, får du 1 point.
- Hvis du ikke sætter nogen krydser, får du 0 point.
- Hvis du sætter kryds ved et forkert svar, får du $-\frac{1}{k-1}$ point, hvor k er antal svarmuligheder.

For en opgave med vægt $v\,\%$ og med n delspørgsmål, hvor du opnår samlet s point, beregnes din besvarelse af opgaven som:

$$\frac{s}{n} \cdot v\,\%$$

Bemærk at det er muligt at få negative point for en opgave.

Opgave 1 (10 %)

I det følgende angiver $\log n$ 2-tals-logaritmen af n.

	Ja	Nej
$n + 3n$ er $O(2n)$?	☐	☐
n^6 er $O(n^5)$?	☐	☐
$n \log n$ er $O(n^2/\log n)$?	☐	☐
$4n^3$ er $O(3n^4)$?	☐	☐
$n^{\frac{1}{2}}$ er $O(n^{\frac{1}{4}})$?	☐	☐
$\sqrt{n}$ er $O(n^{\frac{1}{2}})$?	☐	☐
27^n er $O(n^{27})$?	☐	☐
4^n er $O(2^{3n})$?	☐	☐
$1/n$ er $O(\log n)$?	☐	☐
$\log n$ er $O(\sqrt{n})$?	☐	☐
n er $O(\frac{1}{2}n)$?	☐	☐
n^4 er $O((n^2)^2)$?	☐	☐
$n^{27} + 28$ er $O(n^{54})$?	☐	☐
$n^{27} \cdot n^{13}$ er $O(n^{30})$?	☐	☐
$n/\log n$ er $O(n)$?	☐	☐
$n + \sqrt{n}$ er $O(n\sqrt{n})$?	☐	☐
$n\sqrt{n}$ er $O(n)$?	☐	☐
$7n - 4n$ er $\Omega(9n)$?	☐	☐
$8^{\frac{1}{3}\log n}$ er $\Theta(n)$?	☐	☐
$\frac{n^3}{n^9}$ er $O(\frac{1}{n^2})$?	☐	☐

Opgave 2 (4 %)

Lad v være en knude i et rød-sort søgetræ, og antag der er n elementer i v's venstre undertræ. Hvor mange elementer kan der så maksimalt være i v's højre undertræ, dvs. hvor ubalanceret kan en knude v være i et rød-sort søgetræ?

$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n\sqrt{n})$	$\Theta(n^2)$	$\Theta(2^n)$
☐	☐	☐	☐	☐

Opgave 3 (10 %)

Angiv for hver af nedenstående algoritmer udførselstiden som funktion af n i O-notation.

Algoritme Loop1(n)
  for $i = 1$ **to** n
    $j = i$
    while $j > 1$
      $j = \lfloor j/2 \rfloor$

Algoritme Loop2(n)
  $s = 1$
  for $i = 1$ **to** n
    for $j = i$ **to** n
      $s = s + i$

Algoritme Loop3(n)
  $i = 1$
  $j = 0$
  while $i \leq n$
    $i = i + i$
    while $j < i$
      $j = j + 1$

Algoritme Loop4(n)
  $i = 1$
  $j = n$
  while $i \leq j$
    $i = i + i$
    $j = j - 1$

Algoritme Loop5(n)
  $i = 1$
  $s = 0$
  while $i \leq n$
    for $j = i$ **to** n
      $s = s + 1$
    $i = i + i$

Algoritme Loop6(n)
  $i = 1$
  $j = 1$
  while $i \leq n$
    $j = j + 1$
    $i = i + j$

	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n\sqrt{n})$	$O(\sqrt{n})$	$O(n^3)$	$O((\log n)^2)$
Loop1	☐	☐	☐	☐	☐	☐	☐	☐
Loop2	☐	☐	☐	☐	☐	☐	☐	☐
Loop3	☐	☐	☐	☐	☐	☐	☐	☐
Loop4	☐	☐	☐	☐	☐	☐	☐	☐
Loop5	☐	☐	☐	☐	☐	☐	☐	☐
Loop6	☐	☐	☐	☐	☐	☐	☐	☐

Opgave 4 (4 %)

1	2	3	4	5	6	7	8	9	10	11
12	11	7	9	8	4	2	3	1	5	6

Angiv hvordan ovenstående binære max-heap ser ud efter HEAP-EXTRACT-MAX.

1	2	3	4	5	6	7	8	9	10	
11	9	7	8	6	4	2	3	1	5	☐

1	2	3	4	5	6	7	8	9	10	
6	11	7	9	8	4	2	3	1	5	☐

1	2	3	4	5	6	7	8	9	10	
11	9	7	3	8	4	2	6	1	5	☐

1	2	3	4	5	6	7	8	9	10	
11	9	7	6	8	4	2	3	1	5	☐

Opgave 5 (4 %)

Angiv den binære max-heap efter indsættelse af elementerne 1, 2, 3, 5, 7, 6, og 4 i den givne rækkefølge, startende med den tomme heap.

1	2	3	4	5	6	7	
1	2	3	5	7	4	6	☐
7	5	6	1	3	2	4	☐
7	5	6	1	2	3	4	☐
7	6	5	4	3	2	1	☐

Opgave 6 (4 %)

1	2	3	4	5	6	7	8	9	10
3	4	7	10	1	5	6	9	11	2

Angiv hvordan ovenstående array ser ud efter anvendelsen af Build-Max-Heap.

1	2	3	4	5	6	7	8	9	10	
11	10	7	9	4	5	6	3	1	2	☐
7	10	6	11	2	5	3	9	4	1	☐
11	10	7	9	2	5	6	3	4	1	☐
11	10	7	9	1	5	6	3	4	2	☐

Opgave 7 (4 %)

Betragt Radix-Sort anvendt på nedenstående liste af tal ($d = 5$, $k = 6$).

53661 45325 32661 10601 31325

Angiv den delvist sorterede liste efter at radix-sort har sorteret tallene efter de *tre* mindst betydende cifre.

10601	31325	32661	45325	53661	☐
45325	31325	10601	53661	32661	☐
10601	45325	31325	53661	32661	☐
31325	45325	10601	53661	32661	☐
45325	31325	10601	32661	53661	☐

Opgave 8 (4 %)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
14	7	23	10	4	6	21	11	15	5	13	8	9	17

Angiv resultatet af at anvende PARTITION(A,2,13) på ovenstående array.

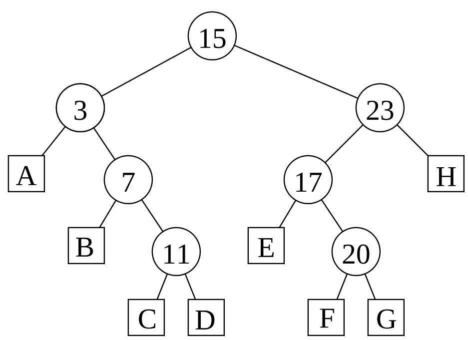
1	2	3	4	5	6	7	8	9	10	11	12	13	14	
14	7	10	4	6	11	5	13	8	9	23	15	21	17	☐

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
14	7	4	6	5	8	9	11	15	23	13	10	21	17	☐

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
14	4	5	6	7	8	9	10	11	13	15	21	23	17	☐

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
14	7	4	6	5	8	9	23	10	21	11	15	13	17	☐

Opgave 9 (4 %)

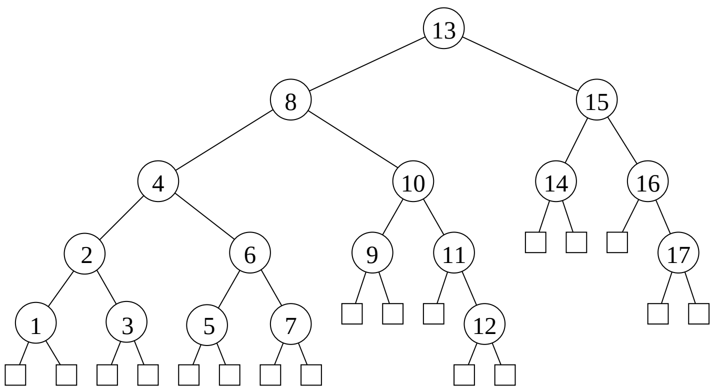


Angiv i hvilke blade A-H i ovenstående ubalancerede binære søgetræ elementerne 9, 21, 16, 12 og 19 skal indsættes.

	A	B	C	D	E	F	G	H
`Insert`(9)	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(21)	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(16)	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(12)	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(19)	☐	☐	☐	☐	☐	☐	☐	☐

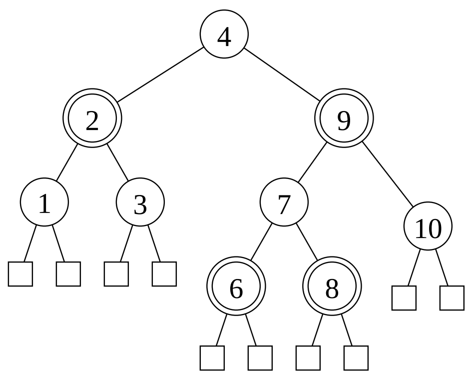
Opgave 10 (4 %)

For hver af nedenstående delmængder, angiv om nedenstående binære træ er et lovligt rød-sort træ hvis netop disse knuder farves røde.



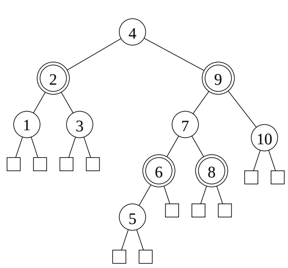
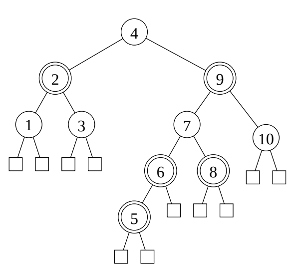
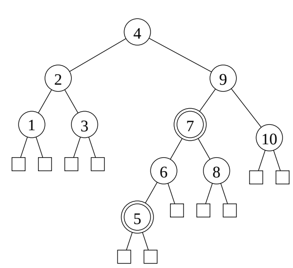
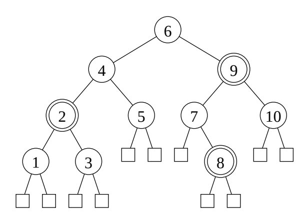
	Ja	Nej
$1, 3, 5, 7, 12, 17$	☐	☐
$1, 3, 4, 5, 7, 9, 12, 17$	☐	☐
$2, 6, 8, 12, 17$	☐	☐
$1, 3, 5, 7, 8, 12, 17$	☐	☐
$1, 3, 5, 7, 8, 11, 16$	☐	☐

Opgave 11 (4 %)



Angiv det resulterende rød-sorte træ når man indsætter 5 i ovenstående rød-sorte træ (dobbeltcirkler angiver røde knuder).

☐ ☐ ☐ ☐

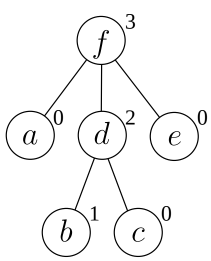
   

Opgave 12 (4 %)

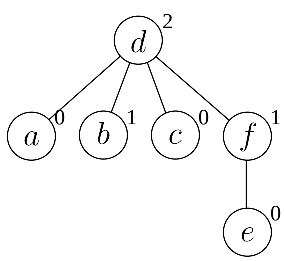
Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der anvendes union-by-rank og stikomprimering.

makeset(a)
makeset(b)
makeset(c)
makeset(d)
makeset(e)
makeset(f)
union(a,b)
union(c,d)
union(e,f)
union(b,c)
union(b,e)
find(a)

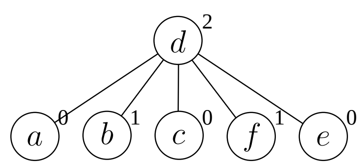
☐



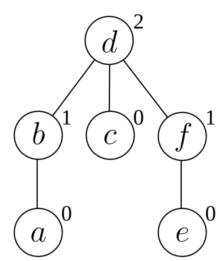
☐



☐



☐



Opgave 13 (4 %)

I følgende hashtabel er anvendt *linear probing* med hashfunktionen $h(k) = 3k \bmod 11$.

0	1	2	3	4	5	6	7	8	9	10
11						2			14	7

Angiv positionerne de tre elementer 3, 5 og 13 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 2, 7, 11 og 14).

	0	1	2	3	4	5	6	7	8	9	10
`Insert`(3)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(5)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
`Insert`(13)	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Opgave 14 (4 %)

I følgende hashtabel af størrelse 7 er anvendt *dobbelt hashing* med hashfunktionerne $h_1(k) = k \bmod 11$ og $h_2(k) = 1 + (k \bmod 6)$.

0	1	2	3	4	5	6
7		20	14			6

Angiv positionerne de tre elementer 2, 10 og 11 vil blive indsat på i hashtabellen (for hver af indsættelserne antager vi at hashtabellen kun indeholder elementerne 6, 7, 14 og 20).

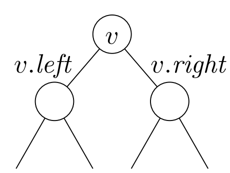
	0	1	2	3	4	5	6
`Insert`(2)	☐	☐	☐	☐	☐	☐	☐
`Insert`(10)	☐	☐	☐	☐	☐	☐	☐
`Insert`(11)	☐	☐	☐	☐	☐	☐	☐

Opgave 15 (4 %)

Givet en streng T indeholdende bogstaver og start- og slut-parenteser `(` og `)`, antages at alle positionerne med parenteser er gemt i et søgetræ, sorteret fra venstre-mod-højre efter stigende position. En knude v gemmer en position $v.p$ og den tilhørende parentes $v.c = T[v.p]$ fra T. For $T =$ ”`a)b(cd(x)dc(a`” gemmes i træet $\langle v.p, v.c\rangle$ parrene $\langle 2,)\rangle, \langle 4, (\rangle, \langle 7, (\rangle, \langle 9,)\rangle$ og $\langle 12, (\rangle$.

Vi ønsker at vedligeholde information om parenteserne er balancerede. I ovenstående eksempel er parenteserne “`)(`$\underline{\texttt{()}}$`(`” ikke balancerede, da kun de markerede parenteser går ud mod hinanden. De restende parenterser “`)((`” vil altid være R `)`-parenteser efterfulgt af L `(`-parenteser, hvor $R \geq 0$ og $L \geq 0$. I eksemplet har vi $R = 1$ og $L = 2$. I en knude v i træet gemmes disse værdier $v.R$ og $v.L$ for delsekvensen af parenteserne i v's undertræ.

Angiv hvorledes $v.R$ kan beregnes når $v.c =$ `)` og R og L værdierne er kendt ved de to børn $v.left$ og $v.right$ (det kan antages at disse begge eksisterer).



$$v.R = \begin{cases} v.left.R + 1 + v.right.R & \square \\ v.left.R + v.right.R + 1 - v.left.L & \square \\ v.left.R + \max\{0, v.right.R + 1 - v.left.L\} & \square \\ v.left.R + 1 + \max\{0, v.right.R - v.left.L\} & \square \end{cases}$$

Transitionssystem ZigZag
Konfigurationer: $\{[i,j] \mid \text{heltal } i \geq 0 \text{ og } j \geq 0\}$

$[i,j] \triangleright [i-1, j+i]$ **if** $i > 0$
$[i,j] \triangleright [i, j-1]$ **if** $j > 0$

Opgave 16 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant for ovenstående transitionssystem ZigZag. Startkonfigurationen antages at være $[n,n]$, hvor $n \geq 0$.

	Ja	Nej
$i \leq n$	☐	☐
$j \leq n$	☐	☐
$i^2 + j^2 \leq 2n^2$	☐	☐
$i^2 + j \leq n^2 + n$	☐	☐
$i(i-1) + j \leq n^2$	☐	☐

Opgave 17 (4 %)

For hver af nedenstående funktioner, angiv om de er en termineringsfunktion for ovenstående transitionssystem ZigZag.

	Ja	Nej
$\mu(i,j) = i + j$	☐	☐
$\mu(i,j) = i \cdot j$	☐	☐
$\mu(i,j) = i^2 + j$	☐	☐
$\mu(i,j) = 2i^2 + j$	☐	☐
$\mu(i,j) = i^2 + j^2$	☐	☐

Opgave 18 (4 %)

Antag, at et givet array $A[1..n]$ repræsenterer en binær max-heap indeholdende n elementer. Hvor hurtigt kan man konstruere et søgetræ (ikke nødvendigvis balanceret) indeholdende elementerne $A[1..n]$, givet at A er en binær max-heap?

$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$
☐	☐	☐	☐

Antag $A[1..n]$ er et sorteret array med n forskellige positive heltal. Lad squares(A) angive antal $A[i]$ hvor $A[i]^2$ også forekommer i A. F.eks. er squares$(1, 3, 4, 7, 9, 16) = 3$, da $1^2 = 1$, $3^2 = 9$ og $4^2 = 16$. Nedenstående algoritme Squares beregner squares(A).

Algoritme Squares($A[1..n]$)
Inputbetingelse : $A[1..n]$ array med n heltal $0 < A[1] < A[2] < \cdots < A[n]$
Outputkrav : $r = \text{squares}(A)$
Metode : $i \leftarrow 1$;
$j \leftarrow 1$;
$r \leftarrow 0$;
$\{I\}$ **while** $j \leq n$ **do**
  if $A[i]^2 < A[j]$ **then** $i \leftarrow i + 1$
  if $A[i]^2 = A[j]$ **then** $i \leftarrow i + 1$; $j \leftarrow j + 1$; $r \leftarrow r + 1$
  if $A[i]^2 > A[j]$ **then** $j \leftarrow j + 1$

Opgave 19 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant I for ovenstående algoritme Squares. Det antages at $A[0] = 0$ og $A[n + 1] = +\infty$.

	Ja	Nej
$1 \leq i \leq j \leq n$	☐	☐
$i \leq j$	☐	☐
$A[i-1]^2 < A[j] \ \wedge \ r = \text{squares}(A[1..j])$	☐	☐
$A[i-1]^2 < A[j] \ \wedge \ r = \text{squares}(A[1..j-1])$	☐	☐
$r = j - 1$	☐	☐

Opgave 20 (4 %)

For hver af nedenstående funktioner, angiv om de er en termineringsfunktion for ovenstående algoritme Squares.

	Ja	Nej
$\mu(n, i, j, r) = i$	☐	☐
$\mu(n, i, j, r) = i + j$	☐	☐
$\mu(n, i, j, r) = j - i$	☐	☐
$\mu(n, i, j, r) = 2(n + 1) - i - j$	☐	☐
$\mu(n, i, j, r) = 2n - i - j$	☐	☐

Opgave 21 (4 %)

Antag at et array $A[1..n]$ indeholder $n \geq 1$ heltal. Positionen i i A siges at være *dominerende* hvis $A[i] > A[j]$ for alle $1 \leq j < i$. Lad $\text{dom}(A)$ angive antal dominerende positioner i i A, hvor $1 \leq i \leq n$. Nedenstående algoritme til venstre beregner $\text{dom}(A)$. For hvert af udsagnene til højre, angiv om de er en invariant I for algoritmen.

Algoritme Domination(A)
Inputbetingelse : Array $A[1..n]$ med $n \geq 1$ heltal
Outputkrav : $r = \text{dom}(A)$
Metode : $i \leftarrow 1$;
$x \leftarrow A[1]$;
$r \leftarrow 1$;
$\{I\}$ **while** $i < n$ **do**
  $i \leftarrow i + 1$;
  if $A[i] > x$ **then**
    $x \leftarrow A[i]$;
    $r \leftarrow r + 1$

	Ja	Nej
$1 \leq i < n$	☐	☐
$x = A[i]$	☐	☐
$x \geq A[i]$	☐	☐
$r = \text{dom}(A[1..n])$	☐	☐
$r = \text{dom}(A[1..i])$	☐	☐

Opgave 22 (4 %)

Betragt en tæller med n cifre $x_0, \ldots, x_{n-1}$, hvor hvert ciffer er et tal fra $\{-1, 0, 1\}$, og hvor værdien af tælleren er $\sum_{i=0}^{n-1} x_i \cdot 2^i$. En tæller kan tælles én op og én ned med operationerne Inc og Dec:

Inc()
  $i \leftarrow 0$
  while $i < n$ **and** $x_i = 1$ **do**
    $x_i \leftarrow 0$
    $i \leftarrow i + 1$
  if $i < n$ **then**
    $x_i \leftarrow x_i + 1$

Dec()
  $i \leftarrow 0$
  while $i < n$ **and** $x_i = -1$ **do**
    $x_i \leftarrow 0$
    $i \leftarrow i + 1$
  if $i < n$ **then**
    $x_i \leftarrow x_i - 1$

Med en passende potentialefunktion kan man argumentere for at Inc og Dec begge tager amortiseret $O(1)$ tid. Angiv for hver af nedenstående om dette er en sådan potentialefunktion Φ.

	Ja	Nej
$\lvert\{i \mid 0 \leq i < n \wedge x_i = 1\}\rvert$	☐	☐
$\lvert\{i \mid 0 \leq i < n \wedge x_i \neq 0\}\rvert$	☐	☐
$\lvert\{i \mid 0 \leq i < n \wedge x_i = 1\}\rvert - \lvert\{i \mid 0 \leq i < n \wedge x_i = -1\}\rvert$	☐	☐
$\sum_{i=0}^{n-1} x_i$	☐	☐
$\sum_{i=0}^{n-1} \lvert x_i \rvert$	☐	☐