Opgave 1 (4 %)

	Ja	Nej
n^7 er $O(n^3)$?	☐	☒
$n(\log n)^3$ er $O(n^2)$?	☒	☐
$n\sqrt{n} + (\log n)^4$ er $O(n^2)$?	☒	☐
2^n er $O(n^2)$?	☐	☒
2^n er $\Omega(3^n)$?	☐	☒

Opgave 2 (4 %)

Opskriv følgende funktioner efter stigende orden med hensyn til O-notationen:

$$7\sqrt{n}$$
$$1/\log n$$
$$\tfrac{1}{2}n^3$$
$$\tfrac{1}{7}n$$
$$(\log n)^4$$

Svar: $1/\log n \quad (\log n)^4 \quad 7\sqrt{n} \quad \tfrac{1}{7}n \quad \tfrac{1}{2}n^3$

Opgave 3 (4 %)

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

Algoritme Loop1(n)
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $j \leftarrow 1$
    while $j \leq n$ **do**
      $j \leftarrow j * 2$
    $i \leftarrow i * 2$

Algoritme Loop2(n)
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $j \leftarrow 1$
    while $j \leq i$ **do**
      $j \leftarrow j + 1$
    $i \leftarrow i * 2$

Algoritme Loop3(n)
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $j \leftarrow 1$
    while $j \leq i$ **do**
      $j \leftarrow j * 2$
    $i \leftarrow i + 1$

Svar Loop1: $O((\log n)^2)$

Svar Loop2: $O(n)$

Svar Loop3: $O(n \log n)$

Opgave 4 (4 %)

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

Algoritme Loop1(n)
  $i \leftarrow 1$
  $s \leftarrow 1$
  while $s \leq n$ **do**
    $i \leftarrow i + 1$
    $s \leftarrow s + i$

Algoritme Loop2(n)
  $i \leftarrow 2$
  while $i \leq n$ **do**
    $i \leftarrow i * i$

Algoritme Loop3(n)
  for $i \leftarrow 1$ **to** n **do**
    $j \leftarrow i$
    while $j \leq n$ **do**
      $j \leftarrow j * 2$

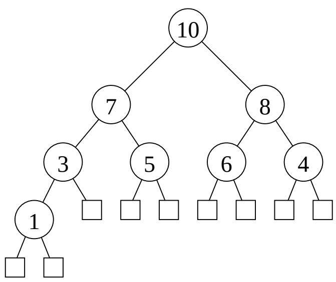
Svar Loop1: $O(\sqrt{n})$

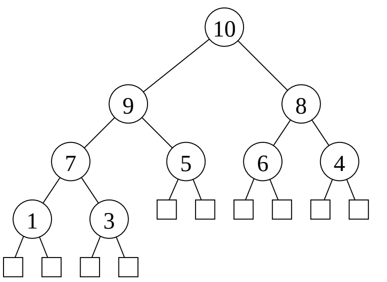
Svar Loop2: $O(\log \log n)$

Svar Loop3: $O(n)$

Opgave 5 (4 %)

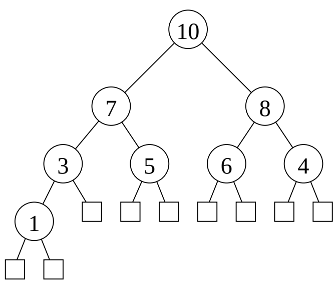
Tegn hvordan nedenstående binære max-heap ser ud efter indsættelse af elementet 9.

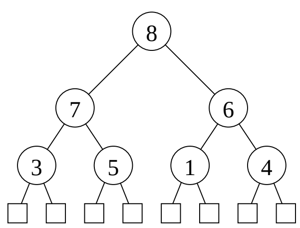




Svar:

Tegn hvordan nedenstående binære max-heap ser ud efter en heap-extract-max operation.

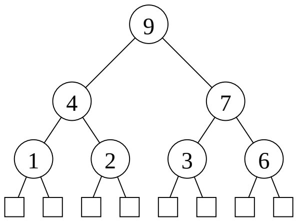




Svar:

Opgave 6 (4 %)

Tegn den binære max-heap efter indsættelse af elementerne 3, 1, 4, 2, 6, 7, 9 i den givne rækkefølge, startende med den tomme heap.

Svar: 

Opgave 7 (4 %)

Angiv hvordan nedenstående array ser ud efter anvendelsen af build-max-heap for arrayet.

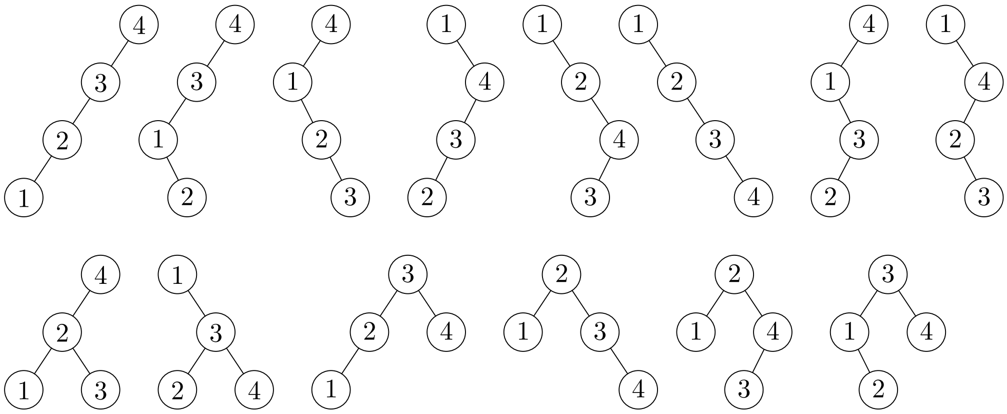
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Svar:

1	2	3	4	5	6	7	8
8	5	7	4	1	6	3	2

Opgave 8 (4 %)

Angiv alle mulige binære søgetræer for mængden $\{1, 2, 3, 4\}$.

Svar: 

Opgave 9 (4 %)

Betragt radix-sort anvendt på nedenstående liste af tal ($d = 5$, $k = 10$). Angiv den delvist sorterede liste efter at radix-sort har sorteret tallene efter de *tre* mindst betydende cifre.

67345 54321 22332 33333 54231 55445 12333

Svar: 54231 54321 22332 33333 12333 67345 55445

Opgave 10 (4 %)

Angiv resultatet af at anvende PARTITION(A,8,16) på nedenstående array.

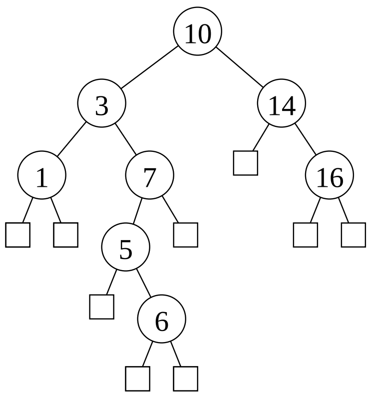
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	8	16	1	6	2	4	13	17	15	3	5	18	9	11	24	12	14	10	7	22

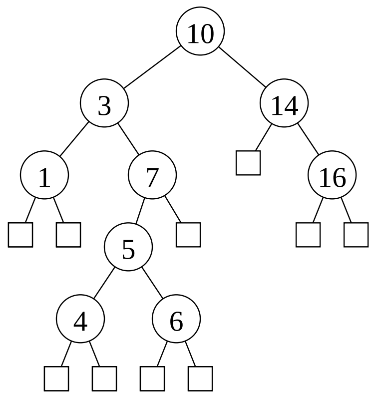
Svar:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	8	16	1	6	2	4	13	3	5	9	11	12	17	15	24	18	14	10	7	22

Opgave 11 (4 %)

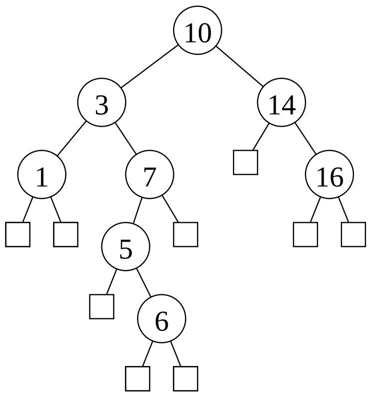
Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter indsættelse af elementet 4.

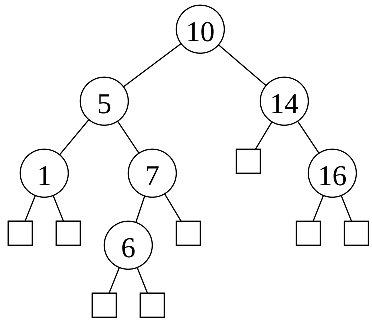




Svar:

Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter slettelse af elementet 3.

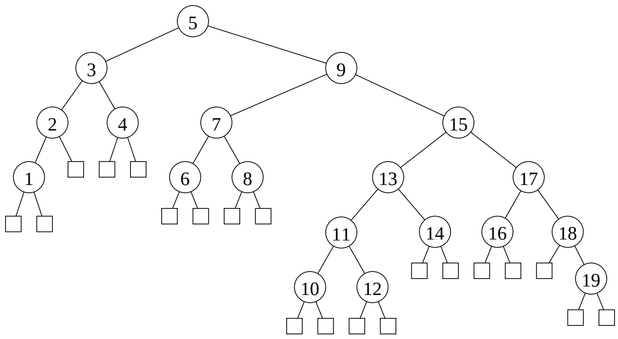


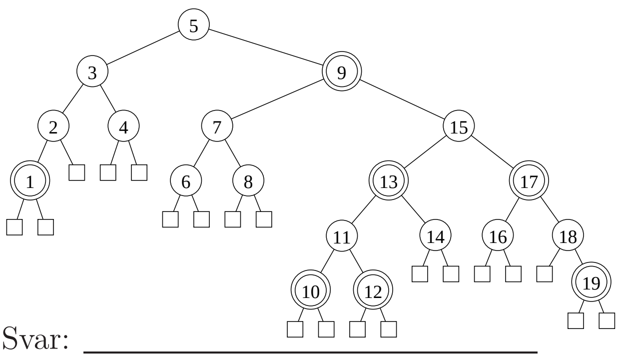


Svar:

Opgave 12 (4 %)

Angiv hvorledes knuderne i nedenstående binære søgetræ kan farves røde og sorte, således at det resulterende træ er et lovligt rød-sort træ.

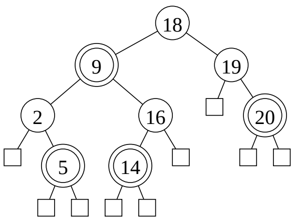


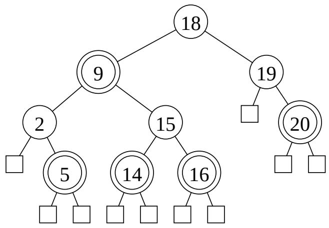


Svar: ______________________________

Opgave 13 (4 %)

Tegn hvordan nedenstående rød-sorte træ (dobbeltcirkler angiver røde knuder) ser ud efter indsættelse af elementet 15.

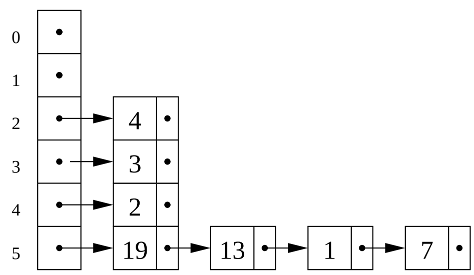




Svar: ______________________________

Opgave 14 (4 %)

Tegn en hashtabel hvor der anvendes kædede lister til at håndtere kollisioner, når hashfunktionen er $h(k) = 5k \bmod 6$ og der indsættes elementerne 3, 2, 7, 4, 1, 13, og 19 i den givne rækkefølge.



Svar: ______________________________

Opgave 15 (4 %)

Tegn hvordan en hashtabel der anvender *linear probing* ser ud efter at elementerne 5, 2, 4, 12, 7, 3, og 0 indsættes i den givne rækkefølge, når hashfunktionen er $h(k) = 3k \bmod 8$.

0	1	2	3	4	5	6	7

0	1	2	3	4	5	6	7
7	3	0		4	12	2	5

Svar: ____

Opgave 16 (4 %)

Tegn hvordan en hashtabel der anvender *quadratic probing* ser ud efter at elementerne 2, 17, 5, 1, og 11 indsættes i den givne rækkefølge, når hashfunktionen er $h(k, i) = 2k + 3i + 4i^2 \bmod 16$.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		17		2		11			1	5					

Svar: ____

Opgave 17 (4 %)

Tegn hvordan en hashtabel der anvender *double hashing* ser ud efter at elementerne 18, 5, 2, 7, og 15 indsættes i den givne rækkefølge, når hashfunktionen er

$$
\begin{aligned}
h(k,i) &= (h_1(k) + i \cdot h_2(k)) \bmod 13 \\
h_1(k) &= 2k \bmod 16 \\
h_2(k) &= 1 + (3k \bmod 7)
\end{aligned}
$$

0	1	2	3	4	5	6	7	8	9	10	11	12

0	1	2	3	4	5	6	7	8	9	10	11	12
	7			18	15					5	2	

Svar: ____

(Opgavesættet fortsætter)

Opgave 18 (4 %)

Betragt en heap implementeret i et array. Overløb håndteres ved at allokere et nyt array af dobbelt størrelse og kopiere indholdet af det gamle array til det nye array. Lad den aktuelle størrelse af arrayet være N og antallet af elementer i heapen n. For hver af nedenstående funktioner angiv om de kan anvendes som en potentiale funktion til at argumenter for at heap-insert og heap-extract-max kræver amortiseret $O(\log n)$ tid.

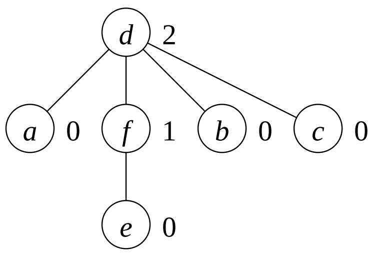
	Ja	Nej
$\Phi(n, N) = n$	☐	☒
$\Phi(n, N) = 2n - N$	☒	☐
$\Phi(n, N) = N - 2n$	☐	☒
$\Phi(n, N) = \log n$	☐	☒
$\Phi(n, N) = \log(N/n)$	☐	☒

(Opgavesættet fortsætter)

Opgave 19 (4 %)

Angiv den resulterende union-find struktur efter nedenstående sekvens af operationer, når der kun 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(a, c)
Union(a, e)



Svar: ______________________________

Opgave 20 (4 %)

Følgende spørgsmål vedrører union-find strukturer, hvor n betegner antallet af elementer i union-find strukturen.

	Ja	Nej
Hvis der anvendes linking-by-rank, så er den maksimale dybde af en union-find struktur $O(\log n)$?	☒	☐
Når der anvendes stiforkortning men ikke linking-by-rank, så er den maksimale dybde af en union-find struktur $O(\log n)$?	☐	☒
Hvis der anvendes linking-by-rank, så har alle børn til en knude forskellige rank ?	☐	☒
Hvis der anvendes linking-by-rank, så har faderen til en knude v altid større rank end v's rank ?	☒	☐
Hvis der anvendes linking-by-rank, så har en union-find struktur højest $\frac{n}{2^i}$ knuder med rank i ?	☒	☐

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

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

Opgave 21 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant for ovenstående transitionssystem CountDown. Startkonfigurationen antages at være $[n, n]$ hvor $n \geq 1$.

	Ja	Nej
$i + j > 0$	☐	☒
$i + j \leq n$	☐	☒
$i \leq j$	☐	☒
$i \cdot j = n^2$	☐	☒
$i \leq n \cdot j$	☐	☒

Opgave 22 (4 %)

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

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

Algoritme Square(n)
Inputbetingelse : heltal $n \geq 2$
Outputkrav : $r = n^2$
Metode : $i \leftarrow 0$
$s \leftarrow 0$
$\{I\}$ **while** $i < n$ **do**
  $i \leftarrow i + 1$
  $s \leftarrow s + i$
$r \leftarrow s + s - i$

Opgave 23 (4 %)

For hver af nedenstående udsagn, angiv om de er en invariant I for ovenstående algoritme Square.

	Ja	Nej
$i \leq s$	☒	☐
$s \leq n$	☐	☒
$s = i^2$	☐	☒
$s = s + i$	☐	☒
$s = \frac{1}{2} \cdot i^2$	☐	☒

Opgave 24 (4 %)

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

	Ja	Nej
$\mu(r, p, n) = i$	☐	☒
$\mu(r, p, n) = n - i$	☒	☐
$\mu(r, p, n) = n - s$	☐	☒
$\mu(r, p, n) = n^2 - s$	☒	☐
$\mu(r, p, n) = (n - i)^2 - s$	☐	☒

(Opgavesættet fortsætter)

Opgave 25 (4 %)

Givet to arrays $A = A[1]A[2]\cdots A[n]$ og $B = B[1]B[2]\cdots B[n]$, beregner nedenstående algoritme prik produktet af de to vektorer. Dette betegnes

$$\mathrm{dot}(A, B) = \sum_{i=1}^{n} A[i] \cdot B[i] \ .$$

For at vise gyldigheden af algoritmen skal I_i og I_p være invarianter omkring variablerne i og p. Angiv invarianter hvormed gyldigheden af algoritmen kan bevises (bevis for invarianterne kræves ikke). Det antages at A, B, og n ikke kan ændres af algoritmen.

Algoritme arrayDot(A, B)
Inputbetingelse : to arrays A og B hver med n heltal
Outputkrav : $p = \mathrm{dot}(A, B)$
Metode : $i \leftarrow 0$
$p \leftarrow 0$
$\{I_i \ \wedge \ I_p\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$
$p \leftarrow p + A[i] \cdot B[i]$

Svar I_i: $0 \leq i \leq n$

Svar I_p: $p = \mathrm{dot}(A[1..i], B[1..i])$

For at kunne bevise at algoritmen terminerer, kræves en passende termineringsfunktion. Angiv en termineringsfunktion (bevis for at termineringsfunktionen har de nødvendige egenskaber kræves ikke).

Svar μ: $n - i$