Opgave 1 (4 %)

	Ja	Nej
n^5 er $O(n^7)$?	☒	☐
$(\log n)^2$ er $O(\sqrt{n})$?	☒	☐
$\sqrt{n}(\log n)^2$ er $O(n)$?	☒	☐
3^n er $O(2^n)$?	☐	☒
n^2 er $\Omega(n^3)$?	☐	☒

Opgave 2 (4 %)

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

$$5n^4$$
$$3\log n$$
$$\sqrt{n}$$
$$4n^5$$
$$n/\log n$$

Svar: $3\log n \quad \sqrt{n} \quad n/\log n \quad 5n^4 \quad 4n^5$

Opgave 3 (4 %)

Angiv for hver af nedenstående summer deres sum som function af n i O-notation:

$$\sum_{i=1}^{n} i = O(n^2)$$

$$\sum_{i=1}^{n} \log i = O(n \cdot \log n)$$

$$\sum_{i=1}^{\log n} 2^i = O(n)$$

$$\sum_{i=1}^{\log n} \frac{n}{2^i} \cdot i = O(n)$$

Opgave 4 (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**
    $j \leftarrow i$
    while $j \leq n$ **do**
      $j \leftarrow j + 1$
    $i \leftarrow i + 1$

Algoritme Loop2(n)
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $j \leftarrow i$
    while $j \geq 0$ **do**
      $j \leftarrow j - 1$
    $i \leftarrow i + 1$

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

Svar Loop1: $O(n^2)$

Svar Loop2: $O(n^2)$

Svar Loop3: $O(\sqrt{n})$

Opgave 5 (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 $i \leq n$ **do**
    $j \leftarrow 1$
    while $j \leq s$ **do**
      $j \leftarrow j + 1$
    $i \leftarrow i + 1$
    $s \leftarrow 2 * s$

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

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

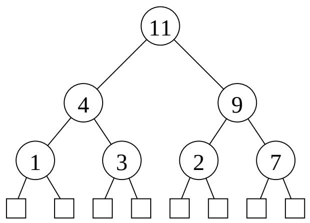
Svar Loop1: $O(2^n)$

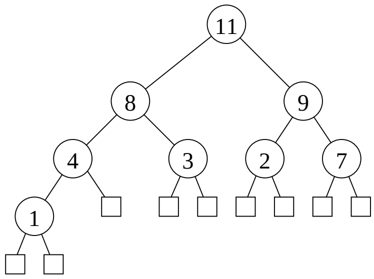
Svar Loop2: $O(n)$

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

Opgave 6 (4 %)

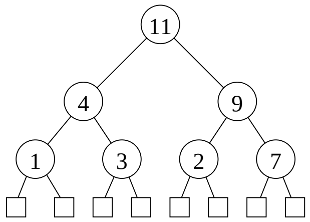
Tegn hvordan nedenstående binære max-heap ser ud efter indsættelse af elementet 8.

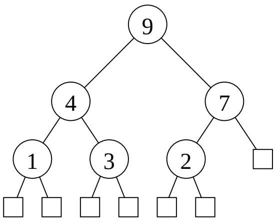




Svar:

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

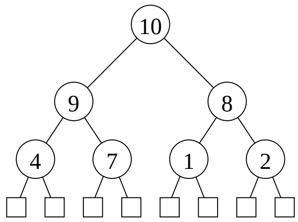




Svar:

Opgave 7 (4 %)

Tegn den binære max-heap efter indsættelse af elementerne 10, 4, 1, 9, 7, 2, 8 i den givne rækkefølge, startende med den tomme heap.



Svar:

Opgave 8 (4 %)

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

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

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

Svar:

Opgave 9 (4 %)

Betragt radix-sort anvendt på nedenstående liste af tal ($d = 4$, $k = 10$). Angiv den delvist sorterede liste efter at radix-sort har sorteret tallene efter de *tre* mindst betydende cifre.

6716 8723 8722 2723 4723 5716 8124 4716

Svar: 8124 6716 5716 4716 8722 8723 2723 4723

Opgave 10 (4 %)

Angiv resultatet af at anvende PARTITION(A,5,13) 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

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

Opgave 11 (4 %)

Er følgende sorteringsalgoritmer stabile:

	Ja	Nej
InsertionSort	☒	☐
MergeSort	☒	☐
RadixSort	☒	☐
QuickSort	☐	☒
HeapSort	☐	☒

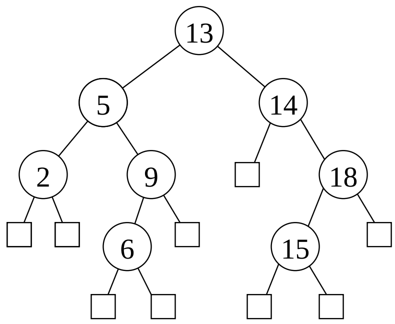
Opgave 12 (4 %)

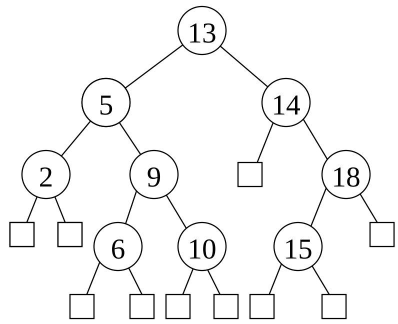
Hvilke af følgende udsagn om rød-sorte træer er sande?

	Ja	Nej
Antal interne røde knuder $\leq$ antal interne sorte knuder i et rød-sort træ	☐	☒
En indsættelse i et rød-sort træ laver worst-case $O(1)$ rotationer	☒	☐
Pladsforbruget af et rød-sort træ med n elementer er $O(\log n)$	☐	☒
En indsættelse i et rød-sort træ omfarver worst-case $O(1)$ knuder	☐	☒
En indsættelse i et rød-sort træ omfarver amortiseret $O(1)$ knuder	☒	☐

Opgave 13 (4 %)

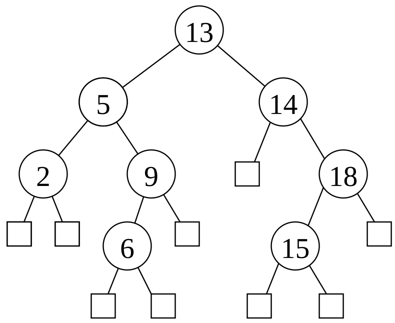
Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter indsættelse af elementet 10.

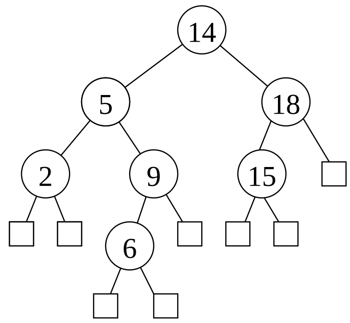




Svar: ______________________________

Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter slettelse af elementet 13.

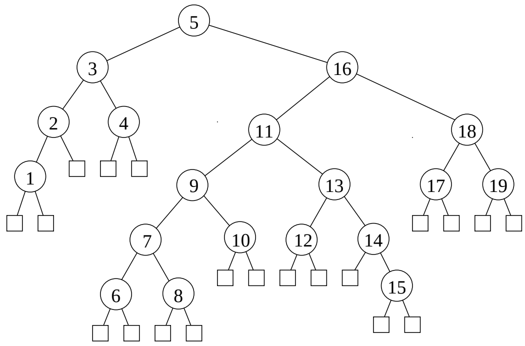


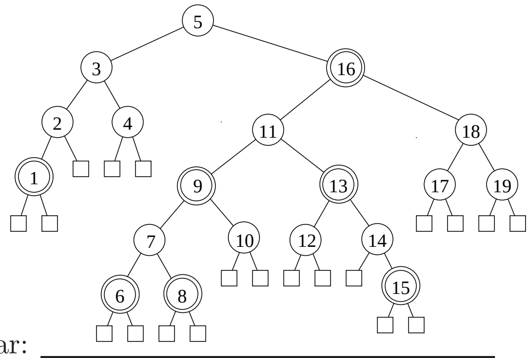


Svar: ______________________________

Opgave 14 (4 %)

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

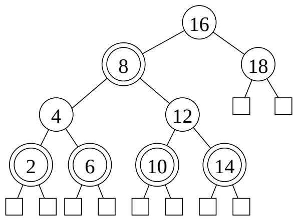




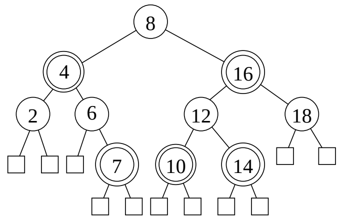
Svar: ______________________________

Opgave 15 (4 %)

Tegn hvordan nedenstående rød-sorte træ (dobbeltcirkler angiver røde knuder) ser ud efter indsættelse af elementet 7.



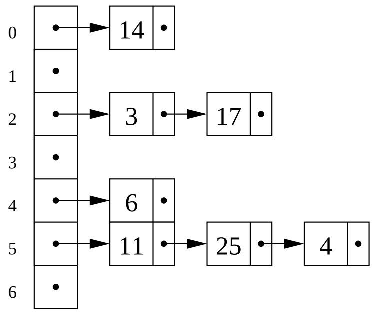
Svar:



Opgave 16 (4 %)

Tegn en hashtabel hvor der anvendes kædede lister til at håndtere kollisioner, når hashfunktionen er $h(k) = 3k \bmod 7$ og der indsættes elementerne 6, 17, 4, 3, 25, 14, og 11 i den givne rækkefølge.

Svar:



Opgave 17 (4 %)

Tegn hvordan en hashtabel der anvender *linear probing* ser ud efter at elementerne 3, 2, 5, 6, 12, 4, og 14 indsættes i den givne rækkefølge, når hashfunktionen er $h(k) = 5k \bmod 9$.

0	1	2	3	4	5	6	7	8

Svar:

0	1	2	3	4	5	6	7	8
14	2	4	6			3	5	12

Opgave 18 (4 %)

Tegn hvordan en hashtabel der anvender *quadratic probing* ser ud efter at elementerne 15, 9, 16, 12, og 2 indsættes i den givne rækkefølge, når hashfunktionen er $h(k, i) = 3k + 7i + 6i^2 \bmod 13$.

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

Svar:

0	1	2	3	4	5	6	7	8	9	10	11	12
	9				2	15			16	12		

(Opgavesættet fortsætter)

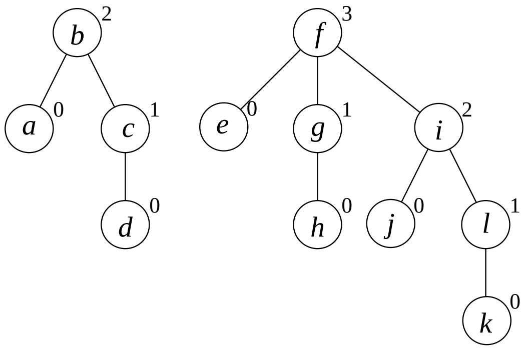
Opgave 19 (4 %)

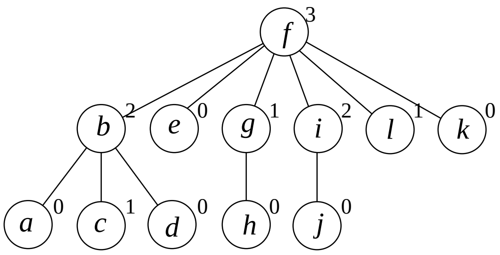
Angiv for hver af nedenstående datastrukturer tiden for at understøtte operationerne Find, Insert, og Delete-Max.

	Find(x)	Insert(x)	Delete-Max()
Max-heap	$O(n)$	$O(\log n)$	$O(\log n)$
Rød-sort søgetræ	$O(\log n)$	$O(\log n)$	$O(\log n)$
Sorteret dobbelt-kædet liste	$O(n)$	$O(n)$	$O(1)$

Opgave 20 (4 %)

Angiv den resulterende union-find struktur efter UNION(d, k), når der anvendes union-by-rank og stikomprimering (tallene angiver knudernes rang).





Svar:

Transitionssystem Frem-og-tilbage
Konfigurationer: $\{[i,j] \mid \text{heltal } i,j \wedge i \geq 0 \wedge j \geq 0\}$

$[i,j] \triangleright [i+1, j-1]$ **if** $j \geq 1$
$[i,j] \triangleright [i-3, j+2]$ **if** $i \geq 3$
$[i,j] \triangleright [i+4, j-3]$ **if** $j \geq 3$

Opgave 21 (4 %)

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

	Ja	Nej
$i + j \geq 0$	☒	☐
$i + j \leq 2n$	☐	☒
$i \leq j$	☐	☒
$5i + 7j \leq 12n$	☒	☐
$i \leq \frac{3}{2}(n - j)$	☐	☒

Opgave 22 (4 %)

For hver af nedenstående funktioner, angiv om de er en termineringsfunktion for ovenstående transitionssystem Frem-og-tilbage.

	Ja	Nej
$\mu(i,j) = i$	☐	☒
$\mu(i,j) = j$	☐	☒
$\mu(i,j) = i + j$	☐	☒
$\mu(i,j) = 5i + 7j$	☒	☐
$\mu(i,j) = 3i + 4j$	☐	☒

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

Opgave 23 (4 %)

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

	Ja	Nej
$i \leq s$	☒	☐
$i \leq n$	☐	☒
$s = i^2$	☒	☐
$s = s + 2i + 1$	☐	☒
$s^2 = n$	☐	☒

Opgave 24 (4 %)

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

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

Opgave 25 (4 %)

Givet et array $A = A[1]A[2]\cdots A[n]$ og et element x, beregner nedenstående algoritme det største element $\leq x$ i arrayet. Denne værdi betegnes

$$\text{Pred}(A, x) = \max\{A[i] \mid 1 \leq i \leq n \wedge A[i] \leq x\} .$$

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

Algoritme Predecessor(A)
Inputbetingelse : array A med n heltal, $n \geq 1$, og et heltal x
Outputkrav : $r = \text{Pred}(A, x)$
Metode : $r \leftarrow -\infty$;
$i \leftarrow 1$;
$\{I_r \wedge I_i\}$ **while** $i \leq n$ **do**
  if $A[i] \leq x$ and $A[i] > r$ **then**
    $r \leftarrow A[i]$;
  $i \leftarrow i + 1$;

Svar I_i: $1 \leq i \leq n + 1$

Svar I_r: $r = \text{Pred}(A[1..i-1], x)$

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: $n + 1 - i$