Opgave 1 (4 %)

	Ja	Nej
n^2 er $\Omega(n)$?	☒	☐
n^3 er $O(2^n)$?	☒	☐
$7n$ er $O(8 \log n)$?	☐	☒
$5+7$ er $O(n)$?	☒	☐
$\sqrt{n}$ er $O(n^{1/3})$?	☐	☒

Opgave 2 (4 %)

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

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

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

Opgave 3 (4 %)

I det følgende angiver f og g positive ikke-aftagende funktioner. Hvilke af følgende udsagn medfører at $f(n)$ er $O(g(n))$?

	Ja	Nej
Der findes $n \in \mathbb{N}$ så $f(n) \leq g(n)$	☐	☒
For alle $n \in \mathbb{N}$ er $f(n) \leq g(n)$	☒	☐
For alle $N \in \mathbb{N}$ findes $n \geq N$ så $f(n) \leq g(n)$	☐	☒
Der findes $N \in \mathbb{N}$ så for alle $n \geq N$ er $f(n) \leq g(n)$	☒	☐
$g(n) = \Omega(f(n))$	☒	☐

Opgave 4 (4 %)

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

Algoritme Loop1(n)
  $i = 1$
  while $i \leq n$
    $i = i + i + i$

Algoritme Loop2(n)
  $i = 1$
  while $i \leq n$
    $j = 1$
    while $j \leq n$
      $j = j + j$
      $i = i + 1$

Algoritme Loop3(n)
  for $i = 1$ **to** n
    $j = 1$
    while $j \leq i$
      $j = j + 1$

Svar Loop1: $O(\log n)$

Svar Loop2: $O(n)$

Svar Loop3: $O(n^2)$

Opgave 5 (4 %)

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

Algoritme Loop1(n)
  $i = 1$
  $j = 1$
  while $i \leq n$
    $j = j + 1$
    $i = i + j$

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

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

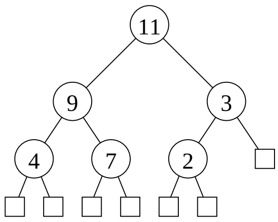
Svar Loop1: $O(\sqrt{n})$

Svar Loop2: $O((\log n)^2)$

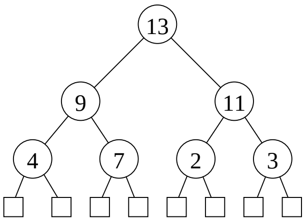
Svar Loop3: $O(n^2)$

Opgave 6 (4 %)

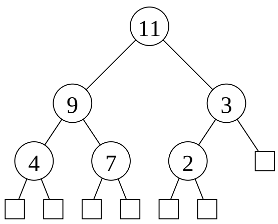
Tegn hvordan nedenstående binære max-heap ser ud efter indsættelse af elementet 13.



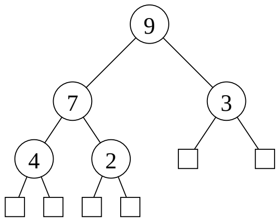
Svar:



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



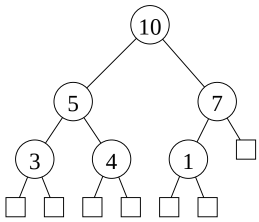
Svar:



Opgave 7 (4 %)

Tegn den binære max-heap efter indsættelse af elementerne 5, 3, 1, 4, 7, 10 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
3	7	2	8	1	4	10	9	6	5

Svar:

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

(Opgavesættet fortsætter)

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 *to* mindst betydende cifre.

4227 1834 4400 0734 1327 9909

Svar: 4400 9909 4227 1327 1834 0734

Opgave 10 (4 %)

Angiv resultatet af at anvende PARTITION(A,3,13) på nedenstående array.

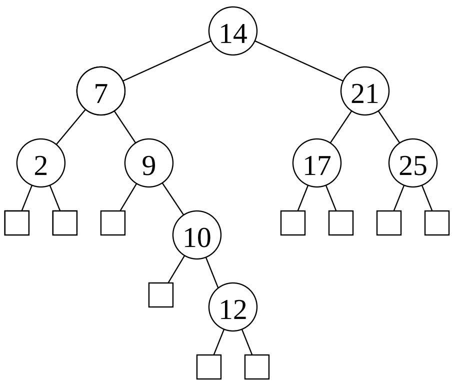
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	8	16	1	6	2	4	13	17	15	3	18	5	9	11	24	12	14	10	7	22

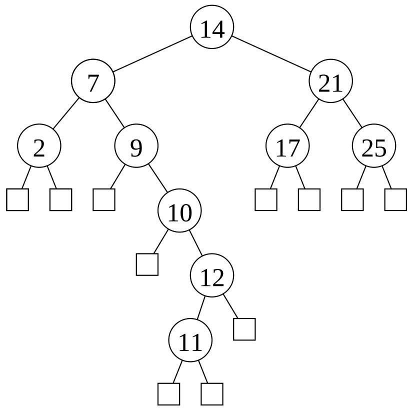
Svar:

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	3	5	9	13	18	17	15	11	24	12	14	10	7	22

Opgave 11 (4 %)

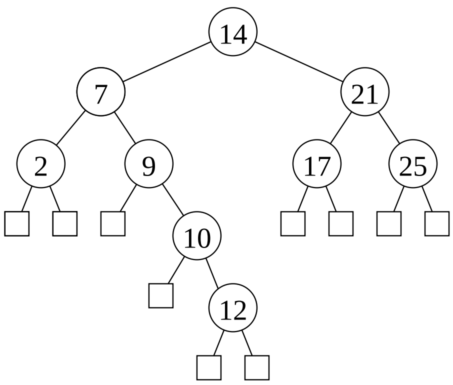
Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter indsættelse af elementet 11.

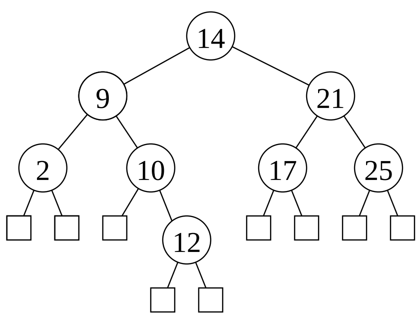




Svar:

Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter slettelse af elementet 7.

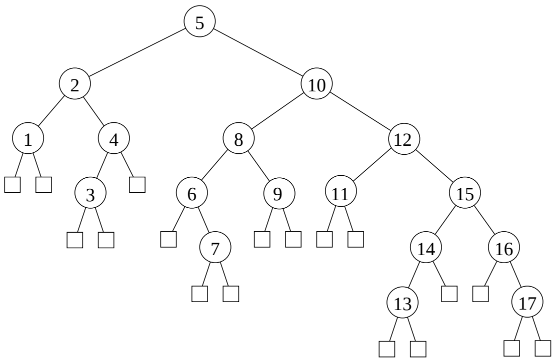




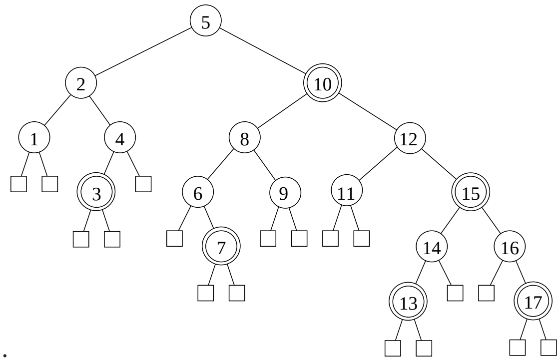
Svar:

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.

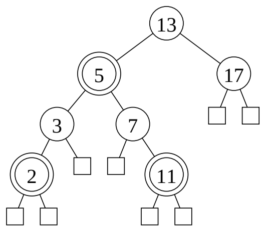


Svar:

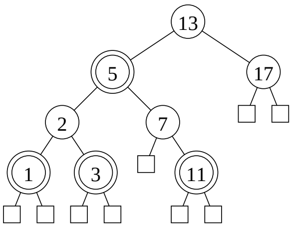


Opgave 13 (4 %)

Tegn hvordan nedenstående rød-sorte træ (dobbeltcirkler angiver røde knuder) ser ud efter indsættelse af elementet 1.



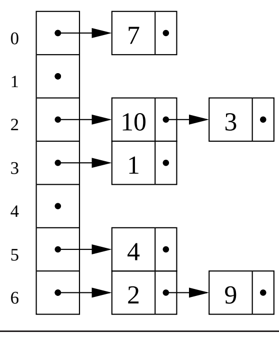
Svar:



Opgave 14 (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 4, 1, 3, 9, 10, 7, og 2 i den givne rækkefølge.

Svar:



Opgave 15 (4 %)

Tegn hvordan en hashtabel der anvender *linear probing* ser ud efter at elementerne 6, 13, 0, 4, 8, 14, 7, og 3 indsættes i den givne rækkefølge, når hashfunktionen er $h(k) = 3k \bmod 10$.

0	1	2	3	4	5	6	7	8	9

0	1	2	3	4	5	6	7	8	9
0	7	4	14	8	3			6	13

Svar: ____________________

Opgave 16 (4 %)

Tegn hvordan en hashtabel der anvender *dobbelt hashing* ser ud efter at elementerne 2, 5, 7, 8, 3, 6, og 1 indsættes i den givne rækkefølge, når hashfunktionerne er $h_1(k) = 2k \bmod 10$ og $h_2(k) = 3k \bmod 10$, og hashtabellen har størrelse 10.

0	1	2	3	4	5	6	7	8	9

0	1	2	3	4	5	6	7	8	9
5		6	3	2	7	8		1	

Svar: ____________________

Opgave 17 (4 %)

Angiv for hver af nedenstående datastrukturer indeholdende n elementer, hvor lang tid det tager at rapportere elementerne i datastrukturen i sorteret rækkefølge, som funktion af n i O-notation.

Rød-sort søgetræ ? Svar: $O(n)$

Binær max-heap ? Svar: $O(n \log n)$

Hashtabel med linear probing ved 50% fyldningsgrad ? Svar: $O(n \log n)$

Opgave 18 (4 %)

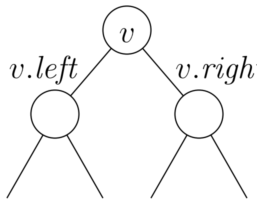
For en sorteret liste af tal $x_1, \ldots, x_n$, definerer vi

$$\mathrm{sep}(x_1, \ldots, x_n) = \min\{x_n - x_{n-1}, x_{n-1} - x_{n-2}, \ldots, x_3 - x_2, x_2 - x_1\}$$

F.eks. er

$$\mathrm{sep}(1, 3, 8, 11, 13, 19) = \min\{19 - 13, 13 - 11, 11 - 8, 8 - 3, 3 - 1\} = 2$$

Betragt et søgetræ hvor hver knude v ud over et element $v.e$, gemmer $v.sep$ som er lig sep(elementerne i v's undertræ), og $v.min$ og $v.max$ som er hhv. det mindste element og største element i v's undertræ. Angiv hvorledes disse værdier kan beregnes når den tilsvarende information er kendt ved de to børn $v.left$ og $v.right$ (det kan antages at disse begge eksisterer). F.eks. betegner $v.right.min$ det mindste element i v's højre undertræ.



Svar $v.min$ = $v.left.min$

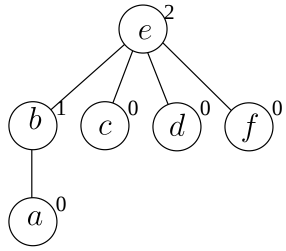
Svar $v.max$ = $v.right.max$

Svar $v.sep$ = $\min\{v.left.sep, v.right.sep, v.e - v.left.max, v.right.min - v.e\}$

Opgave 19 (4 %)

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

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



Svar:

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

$[i, j] \triangleright [i/2, j * j]$ **if** $i \geq 1 \wedge i$ lige
$[i, j] \triangleright [i - 1, j]$ **if** $i \geq 1$
$[i, j] \triangleright [i, j - 1]$ **if** $j \geq 1$

Opgave 20 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant for ovenstående transitionssystem Halver-og-kvadrer. Startkonfigurationen antages at være $[n, 2]$ hvor $n \geq 0$.

	Ja	Nej
$i \leq n$	☒	☐
$j \leq i$	☐	☒
$j \leq n$	☐	☒
$j^i \leq 2^n$	☒	☐
$i + j \leq n + 2$	☐	☒

Opgave 21 (4 %)

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

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

Algoritme Loglog(n)
Inputbetingelse : heltal $n \geq 2$
Outputkrav : $r = \lfloor \log_2(\log_2(n)) \rfloor$
Metode : $p \leftarrow 2$
$r \leftarrow 0$
$\{I\}$ **while** $p * p \leq n$ **do**
    $r \leftarrow r + 1$
    $p \leftarrow p * p$

Opgave 22 (4 %)

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

	Ja	Nej
$p \geq r$	☒	☐
$p = 2^{2^r}$	☒	☐
$p = 2^r$	☐	☒
$n - r = p$	☐	☒
$p \leq n$	☒	☐

Opgave 23 (4 %)

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

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

Opgave 24 (4 %)

Givet et sorteret array A med n tal, $n \geq 1$, beregner nedenstående algoritme antal forskellige elementer i A. Denne værdi betegnes

$$\text{SetSize}(A) = 1 + |\{i \mid A[i-1] \neq A[i] \wedge 1 < i \leq n\}|$$

For at vise gyldigheden af algoritmen skal I_i og I_s være invarianter omkring variablerne i og s. Angiv invarianter hvormed gyldigheden af algoritmen kan bevises (bevis for invarianterne kræves ikke).

Algoritme ComputeSetSize(n)
Inputbetingelse : sorteret array A med n tal, $n \geq 1$
Outputkrav : $s = \text{SetSize}(A)$
Metode : $i \leftarrow 1$;
$s \leftarrow 1$;
$\{I_i \ \wedge \ I_s\}$ **while** $i < n$ **do**
  if $A[i] \neq A[i+1]$ **do**
    $s \leftarrow s + 1$;
  $i \leftarrow i + 1$;

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

Svar I_s: $s = \text{SetSize}(A[1..i])$

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: $n - i$

Opgave 25 (4 %)

Antag en kø implementeres ved to stakke Front og Tail, og at operationerne på køen er implementeret som:

Algoritme Enqueue(x)
Metode : Tail.push(x)

Algoritme Dequeue()
Metode : **if** Front.empty() **then**
  while not Tail.empty() **do**
    Front.push(Tail.pop())
return Front.pop()

For at argumentere at operationerne tager amortiseret $O(1)$ tid kræves en potentiale funktion. Angiv en potentiale funktion hvorved tiden kan bevises. Argumentation for tiden kræves ikke.

Svar Φ(Front,Tail) : |Tail|