Opgave 1 (4 %)

	Ja	Nej
n^2 er $O(\frac{1}{2}n^4)$?	☒	☐
$\sqrt{n}$ er $O(\log^7 n)$?	☐	☒
2^n er $O(n^4)$?	☐	☒
$3n^2 + 47$ er $O(60n)$?	☐	☒
$n \log n$ er $O(n^{3/2})$?	☒	☐

Opgave 2 (4 %)

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

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

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

Opgave 3 (4 %)

Nedenstående spørgsmål vedrører Randomized Quicksort på input af størrelse n.

	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$
Worst-case antal kald til Partition proceduren	☒	☐	☐
Forventede antal kald til Partition proceduren	☒	☐	☐
Worst-case tid for Randomized Quicksort	☐	☐	☒
Forventede tid for Randomized Quicksort	☐	☒	☐

(Opgavesættet fortsætter)

Opgave 4 (4 %)

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

```
Algoritme Loop1(n)
  i = 1
  while i ≤ n
    i = 3 * i
```

```
Algoritme Loop2(n)
  i = 0
  while i ≤ n
    j = 0
    while j ≤ i
      j = j + 1
    i = i + 1
```

```
Algoritme Loop3(n)
  s = 0
  i = 0
  while s ≤ n
    s = s + i
    i = i + 1
```

Svar Loop1: $O(\log n)$

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 = 1
  while i ≤ n
    j = 1
    while j ≤ i
      j = j * 2
    i = i * 2
```

```
Algoritme Loop2(n)
  for i = 0 to n
    j = 0
    s = 0
    while s ≤ i
      j = j + 1
      s = s + j
```

```
Algoritme Loop3(n)
  i = 0
  s = 0
  q = 0
  while q ≤ n
    i = i + 1
    s = s + i
    q = q + s
```

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

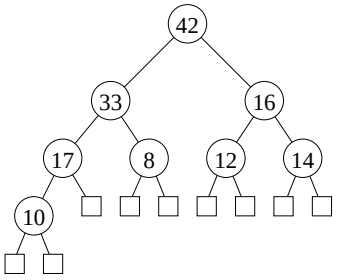
Svar Loop2: $O(n\sqrt{n})$

Svar Loop3: $O(\sqrt[3]{n})$

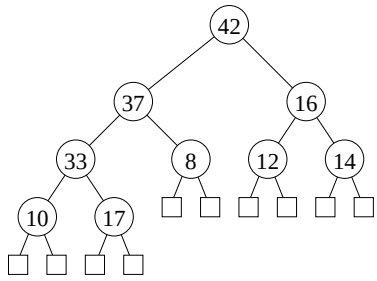
(Opgavesættet fortsætter)

Opgave 6 (4 %)

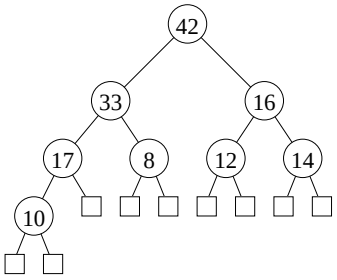
Tegn hvordan nedenstående binære max-heap ser ud efter indsættelse af elementet 37.



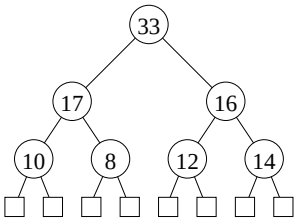
Svar:



Tegn hvordan nedenstående binære max-heap ser ud efter en HEAP-EXTRACT-MAX operation.



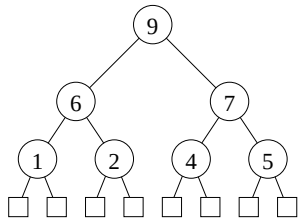
Svar:



Opgave 7 (4 %)

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

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

Svar:

(Opgavesættet fortsætter)

Opgave 9 (4 %)

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

21224 54123 43123 10224 32123

Svar: 54123 43123 32123 21224 10224

Opgave 10 (4 %)

Angiv resultatet af at anvende PARTITION(A,2,14) på nedenstående array.

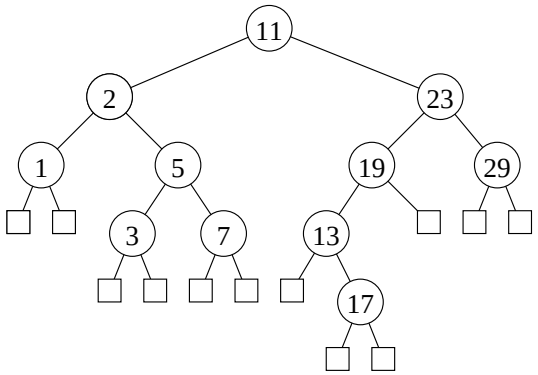
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	1	16	2	15	3	14	4	13	5	12	6	11	7	9	10

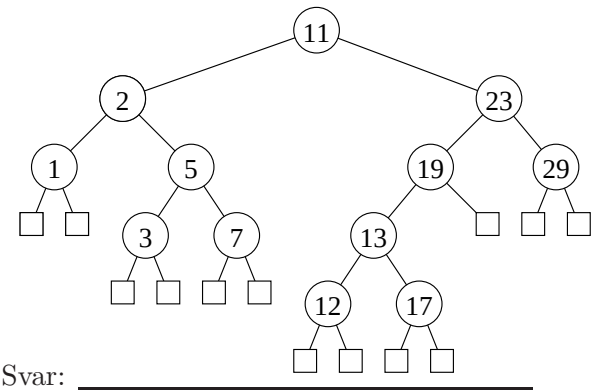
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**16**	**12**	**14**	**11**	**15**	**13**	10

Svar:

Opgave 11 (4 %)

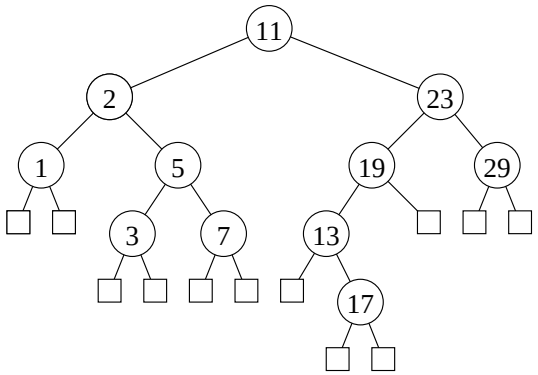
Tegn hvordan nedenstående ubalancerede binære søgetræ ser ud efter indsættelse af elementet 12.

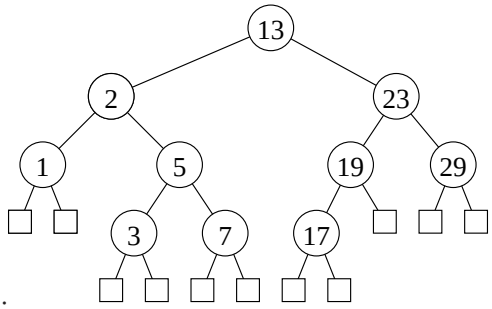




Svar:

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



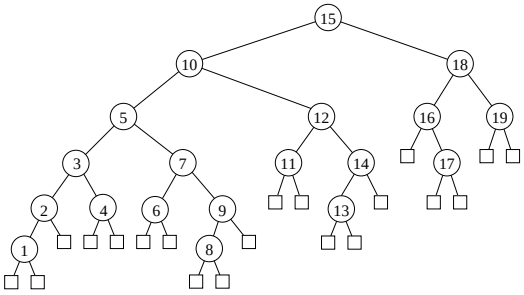


Svar:

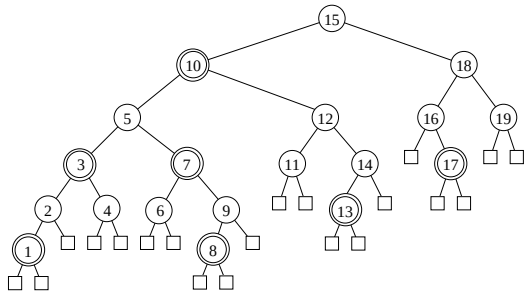
(Opgavesættet fortsætter)

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.

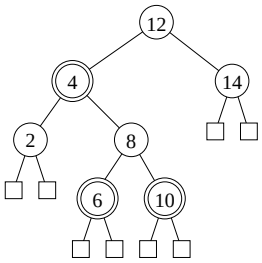


Svar:

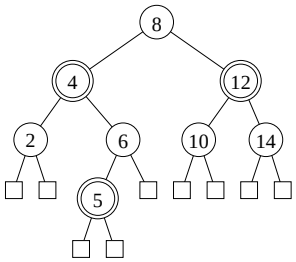


Opgave 13 (4 %)

Tegn hvordan nedenstående rød-sorte træ (dobbeltcirkler angiver røde knuder) ser ud efter indsættelse af elementet 5.



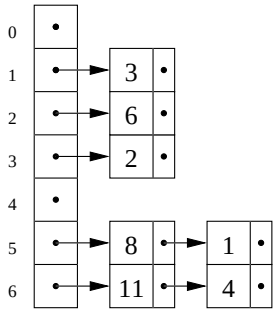
Svar:



Opgave 14 (4 %)

Tegn en hashtabel hvor der anvendes kædede lister til at håndtere kollisioner, når hash-funktionen er $h(k) = 5k \bmod 7$ og der indsættes elementerne 3, 1, 4, 8, 6, 11, og 2 i den givne rækkefølge.

Svar:



(Opgavesættet fortsætter)

Opgave 15 (4 %)

Tegn hvordan en hashtabel der anvender *linear probing* ser ud efter at elementerne 9, 7, 2, 5, og 3 indsættes i den givne rækkefølge, når hashfunktionen er $h(k) = 3k \bmod 7$.

0	1	2	3	4	5	6

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

Svar:

Opgave 16 (4 %)

Tegn hvordan en hashtabel der anvender *dobbelt hashing* ser ud efter at elementerne 7, 1, 2, 0, og 11 indsættes i den givne rækkefølge, når hashfunktionerne er $h_1(k) = 3k \bmod 11$, $h_2(k) = 1 + (2k \bmod 5)$, og hastabellen har størrelse 7.

0	1	2	3	4	5	6

0	1	2	3	4	5	6
0		11	7	2		1

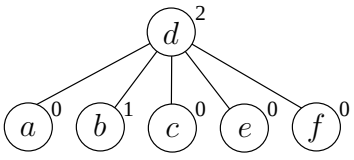
Svar:

Opgave 17 (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(c,d)
union(a,c)
union(b,e)
union(a,f)

Svar:



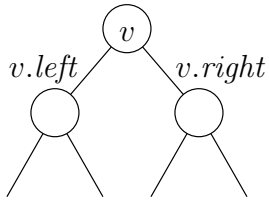
Opgave 18 (4 %)

Betragt en liste af n par $(x_1, w_1), \ldots, (x_n, w_n)$, hvor x_i og w_i er reelle tal og $x_1 < x_2 < \cdots < x_n$. For $1 \leq k \leq n$ udgør $(x_1, w_1), \ldots, (x_k, w_k)$ et præfiks af listen, og vi definerer vægten af præfikset til at være $w_1 + \cdots + w_k$. Vi ønsker at vedligeholde den maksimale præfiksvægt for listen, dvs. maksimum taget over alle præfiksernes vægt.

F.eks. har listen $(1, 4), (2, -3), (5, 8), (7, -2)$ maksimal præfiksvægt $4 + (-3) + 8 = 9$.

Betragt et søgetræ hvor hver knude v gemmer et par $v.x$ og $v.w$, og knuderne er ordnet venstre-mod-højre efter stigende $v.x$. Derudover gemmes i v også $v.sum$ og $v.maxp$, som er hhv. summen af alle vægtene i v's undertræ, og den maksimale præfiksvægt af dellisten gemt 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).

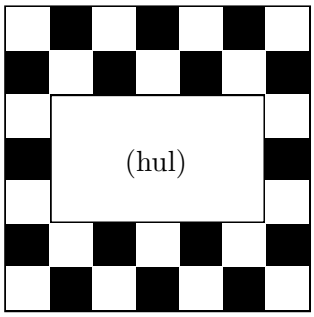


Svar $v.sum$ = $v.left.sum + v.w + v.right.sum$

Svar $v.maxp$ = $\max\{v.left.maxp, v.left.sum + v.w, v.left.sum + v.w + v.right.maxp\}$

Opgave 19 (4 %)

Betragt højrestående 7×7 skakbræt med et 3×5 hul. Vi ønsker at dække de resterende 34 felter med dominobrikker.



For at bevise at skakbrættet ikke kan dækkes med dominobrikker, betragter vi følgende transitionssystem, der indfanger antallet af ikke-dækkede sorte (s) og hvide (h) felter:

$$[s, h] \rhd [s - 1, h - 1]$$

Angiv en invariant, hvorfra det følger at de 34 felter ikke kan dækkes med dominobrikker.

Invariant: $s < h$

(Opgavesættet fortsætter)

Transitionssystem GCD
Konfigurationer: $\{[m,n] \mid \text{heltal } m,n \ \wedge\ m \geq 0 \ \wedge\ n \geq 0 \ \wedge\ m+n \geq 1\}$

$[m,n] \rhd [n,m]$ **if** $m < n$
$[m,n] \rhd [m-n,n]$ **if** $m \geq n \ \wedge\ n > 0$

Opgave 20 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant for ovenstående transitionssystem GCD. Startkonfigurationen antages at være $[n_0, m_0]$, hvor $n_0 \geq 1$ og $m_0 \geq 1$. I det følgende betegner $\gcd(x,y)$ den største fælles divisor i to heltal x og y.

	Ja	Nej
$n \neq 0$	☐	☒
$\gcd(m,n) = \gcd(m_0,n_0)$	☒	☐
$m+n = m_0+n_0$	☐	☒
$m+n \geq 1$	☒	☐
$m \cdot n \geq \gcd(m_0,n_0)$	☐	☒

Opgave 21 (4 %)

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

	Ja	Nej
$\mu(m,n) = m+n$	☐	☒
$\mu(m,n) = 2m+n$	☐	☒
$\mu(m,n) = m+2n$	☒	☐
$\mu(m,n) = m$	☐	☒
$\mu(m,n) = 2n-m$	☐	☒

(Opgavesættet fortsætter)

Hvis et array $A[1..n]$ indeholder et element $A[j]$, der forekommer flere end $n/2$ gange i A, så siges $A[j]$ at være et *majoritetselement*.

Algoritme Majoritet($A[1..n]$)
Inputbetingelse : Array $A[1..n]$ med n heltal, hvor $n \geq 1$ og ét tal forekommer $> n/2$ gange i A
Outputkrav : j, hvor $A[j]$ er majoritetselementet
Metode :

```
i ← 1;
j ← 1;
c ← 1;
{I} while i < n do
        i ← i + 1;
        if c = 0 then
          j ← i;
          c ← 1
        else if A[i] = A[j] then
          c ← c + 1
        else
          c ← c − 1
```

Opgave 22 (4 %)

For hvert af nedenstående udsagn, angiv om de er en invariant I for ovenstående algoritme Majoritet. For $1 \leq k \leq i$, lad $\text{count}(k, i)$ betegne antal forekomster af $A[k]$ i $A[1..i]$, dvs. $\text{count}(k, i) = |\{\ell \mid 1 \leq \ell \leq i \ \wedge \ A[\ell] = A[k]\}|$.

	Ja	Nej
$c = \text{count}(j, i)$	☐	☒
$c \leq \text{count}(j, i)$	☒	☐
$c = i - j$	☐	☒
$2 \cdot \text{count}(k, i) + c \leq i$, for $A[k] \neq A[j]$	☒	☐
$2 \cdot \text{count}(j, i) - c \leq i$	☒	☐

Opgave 23 (4 %)

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

	Ja	Nej
$\mu(n, i, j, c) = i$	☐	☒
$\mu(n, i, j, c) = n - i - c$	☐	☒
$\mu(n, i, j, c) = n - i$	☒	☐
$\mu(n, i, j, c) = i - j$	☐	☒
$\mu(n, i, j, c) = 2(n - i) - c$	☐	☒

(Opgavesættet fortsætter)

Opgave 24 (4 %)

Nedenstående algoritme er en variation af binær søgning efter et element x i et sorteret array $A[1..n]$, hvor man ikke nødvendigvis vælger det midterste element i et delarray $A[L..H]$ at sammenligne x med, men vælger et $A[m]$ afhængig om værdien af x ligger tættest på $A[L]$ eller $A[H]$. For at vise gyldigheden af algoritmen skal $I_{L,H}$ og I_A være invarianter omkring variablerne L og H, og sammenhængen med A. Angiv invarianter hvormed gyldigheden af algoritmen kan bevises (bevis for invarianterne kræves ikke).

Algoritme InterpolationSearch($A[1..n], x$)
Inputbetingelse : Sorteret array $A[1..n]$ med n heltal og et heltal x, hvor $n \geq 1$
Outputkrav : r, hvor $r = -1$ hvis $x \notin A$, ellers $A[r] = x$
Metode :
```
r ← −1;
L ← 1;
H ← n;
{I} while r = −1 ∧ L ≤ H do
        m ← min{H, max{L, L + ⌊((x−A[L])(H−L))/(A[H]−A[L])⌋}};
        if x = A[m] then
          r ← m
        else if x < A[m] then
          H ← m − 1
        else
          L ← m + 1
```

Svar $I_{L,H}$: $1 \leq L \leq H + 1 \leq n + 1$

Svar I_A: $x \notin A[1..L-1] \cup A[H+1..n] \ \wedge \ (r = -1 \ \vee \ A[r] = x)$

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: $H - L + n - r$

Opgave 25 (4 %)

Antag at en binær max-heap med n elementer er gemt i de første n indgange i et array af størrelsen N. Operationerne INSERT og EXTRACT-MAX implementeres som for en standard binær max-heap, på nær INSERT når $n = N$, hvor heapen først kopieres over i et nyt array af dobbelt størrelse $2N$, dvs. N fordobles, hvorefter INSERT operationen udføres som for en standard binær max-heap. Med en passende potentialefunktion kan man argumentere for at INSERT tager amortiseret $O(\log n)$ tid og EXTRACT-MAX tager amortiseret $O(1)$ tid. Angiv en sådan potentialefunktion.

Svar $\Phi =$ $|2n - N| + n \log n$